Topics in Optimisation

David Kirszenblat

Submitted in total fulfilment of the requirements of
the degree of Doctorate of Philosophy
August 2018
Department of Mathematics and Statistics
The University of Melbourne

Abstract

This thesis addresses four problems in continuous and discrete optimisation. The first problem is about using column generation – an advanced technique in mixed integer programming – to design schedules for students in the Royal Australian Navy who are learning to fly helicopters. Selected results were published in the proceedings of the International Congress on Modelling and Simulation. The second problem is about shortest curvature-constrained paths in the plane, i.e., Dubins curves, which find applications in, for example, path planning for unmanned aerial vehicles. We use geometric arguments to produce a classification of length minimising curvature-constrained paths. The results were published in Communications in Analysis and Geometry. The third problem is about the construction of minimal curvature-constrained networks with applications in the design of underground mines. This problem is a novel combination of two classical problems in optimisation, namely, the Steiner problem and the Dubins problem. We give two algorithms for the construction of "Dubins trees" in the plane and in 3D space. The results were selected for publication in a special issue of Journal of Global Optimization on "Distance Geometry: Theory and Applications". The fourth problem is the polynomial Hirsch conjecture. The Hirsch conjecture, posed in 1957, states that the least number edges required to join any pair of vertices of a d-dimensional polyhedron with n facets, i.e., its diameter, is bounded from above by $n - d$. The conjecture is of relevance to the efficiency of the simplex method, a path following method for the solution of linear programming problems. The version of the conjecture for polytopes was disproved by Santos in 2010. However, the question of whether there exists an upper bound for the diameter that is polynomial in d and n remains open. We propose two approaches for tackling this problem. First, we provide an algorithm that allows us to enumerate polytopes and verify a variant of the Hirsch conjecture, called the d-step conjecture, in low dimensions. Second, we provide a method for constructing higher dimensional polyhedra that satisfy the Hirsch bound from arbitrary polyhedra. The method allows us to take an arbitrary linear programming problem and construct an equivalent linear programming problem for which the best possible sequence of pivots to an optimal solution is small.

Declaration

This is to certify that

(i) the thesis comprises only my original work towards the PhD except where indicated in the Preface;

(ii) due acknowledgement has been made in the text to all other material used,

(iii) the thesis is less than 50,000 words in length, exclusive of tables, maps, bibliographies and appendices.

David Kirszenblat

Preface

This thesis addresses four problems in optimisation. Some of the problems are in discrete optimisation, whereas others are in continuous optimisation; some of the problems concern applications, whereas others are more theory oriented. The first three chapters include material from published journal articles. There is a separate introduction to each problem.

Chapter 1 is about using mixed integer programming to solve a scheduling problem for the Australian Defence Force. The bulk of the project was carried out during a six-month Australian Postgraduate Research internship with Defence Science and Technology Group and was funded by the Australian Mathematical Sciences Institute. This is joint work with Brendan Hill, Vicky Mak-Hau, Bill Moran, Vivian Nguyen and Ana Novak, all members of the Algora Systems Team. Part of the work was published in the conference proceedings of the 22nd International Congress on Modelling and Simulation and was presented in Hobart, Tasmania, Australia in December 2017.

- D. Kirszenblat, B. Hill, V. Mak-Hau, B. Moran, V. Nguyen, A. Novak (2017). *Using column generation to solve an aircrew training timetabling problem.* MODSIM, Hobart, Tasmania, Australia, 3 to 8 December 2017

Chapter 2 is about shortest curvature-constrained paths in the plane. The chapter focuses on a relatively small yet crucial contribution to a joint work with José Ayala and Hyam Rubinstein of the University of Melbourne. The work was published in Communications in Analysis and Geometry in 2018 and can be found in the appendix.

- J. Ayala, D. Kirszenblat, and J.H. Rubinstein, *A geometric approach to shortest bounded curvature paths*, Communications in Analysis and Geometry, Vol. 26, No 4, 2018.

Chapter 3 is about shortest curvature-constrained networks. This is joint work with Kash Sirinanda, Marcus Brazil, Peter Grossman, Hyam Rubinstein and Doreen Thomas of the University of Melbourne. We published a paper in Journal of Global Optimization which is to be moved from a regular issue to a special issue on "Distance Geometry: Theory and Applications".

- Kirszenblat, D. & Sirinanda, K. & Brazil, M. & Grossman, P. & Rubinstein, H. & Thomas, D. (2018). *Minimal curvature-constrained networks.* Journal of Global Optimization. 10.1007/s10898-018-0625-2.

Chapters 4 – 6 are dedicated to the polynomial Hirsch conjecture. Chapter 4 reviews the literature while Chapters 5 and 6 explore two different approaches to tackling this problem.

Acknowledgements

I would like to thank my supervisor, Professor J. Hyam Rubinstein, for generously sharing his time and expertise and for challenging me to work on hard problems.

Contents

List of Figures

Chapter 1

Using column generation to solve an aircrew training timetabling problem

1.1 Introduction

This chapter concerns a classical optimisation problem, involving mixed integer programming, that was tackled during a six-month internship at Defence Science and Technology Group, an R&D organisation that is part of Australia's Department of Defence. The goal of the project was to design an efficient algorithm for generating course schedules for students in the Royal Australian Navy who are learning to fly helicopters. The topic of automated schedule and timetable generation is vast and dates back to at least as early at the 1960's (see, e.g., [3, 5]). There is even an International Timetabling Competition providing benchmark data for testing algorithms (http://www.cs.qub.ac.uk/itc2007). In our case, the challenge was to minimise the total time to graduate all students in the program. This benefits the Defence Force, because it saves money. It also benefits the students, who want to graduate as quickly as possible in order to start flying. Each course schedule has to satisfy the following three rules. First, a schedule has to cover all courses in a students' syllabus. Second, a student cannot take two courses at once. Third, a student can take a course only if he or she has already completed its prerequisites. The outcome of the project was a column generation algorithm that allowed for a dramatic reduction in the amount of computation time required to solve this problem [13]. This is important, because we need to be about to quickly and easily generate new schedules at short notice and conduct what-if analysis with different scenarios. Whereas the previous approaches could take more than 24 hours to find a feasible solution if at all, the new algorithm takes less than 30 minutes to find an

optimal solution. The preliminary results of the project are summarised in the following conference paper which was presented at the 22nd International Congress on Modelling and Simulation in Hobart, Australia [31]. At the end of the paper, I discuss some modifications made to the problem formulation while implementing the algorithm for the solution of real-world problems.

1.2 Using column generation to solve an aircrew training timetabling problem

22nd International Congress on Modelling and Simulation, Hobart, Tasmania, Australia, 3 to 8 December 2017
mssanz.org.au/modsim2017

Using column generation to solve an aircrew training timetabling problem

D. Kirszenblat [a], **B. Hill,** [a] **V. Mak-Hau** [b], **B. Moran** [c], **V. Nguyen** [d], **A. Novak** [d]

[a]*School of Mathematics and Statistics, University of Melbourne, Parkville, VIC 3010, Australia*
[b]*School of Information Technology, Deakin University, Waurn Ponds, VIC 3216, Australia*
[c]*Royal Melbourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000, Australia*
[d]*Defence Science and Technology Group, Department of Defence, 506 Lorimer St, Fishermans Bend, VIC 3207, Australia*
Email: d.kirszenblat@student.unimelb.edu.au

Abstract: The Training Authority Aviation (TA-Avn) is an organisation within the Royal Australian Navy (RAN) responsible for managing aviation-specific training for all RAN personnel, who are to be employed in an aviation-related job category. In a temporal sense, the bulk of aircrew training consists of a sequence of major, structured courses and a number of mandatory short courses for which the prerequisite requirements are less strict. Both short and long courses are run repeatedly throughout a year with a fixed number of repetitions and are subject to high and extremely variable course pass rates. It is important to have an ability to quickly and easily regenerate a new timetable at short notice, potentially on a weekly basis depending on whether students have to repeat failed short courses.

In previous work we explored a number of approaches including a stochastic approach to optimisation. In this paper, we adopt a different methodology, using more conventional integer linear programming techniques, specifically, column generation. The problem of designing feasible schedules is formulated as a network flow problem that encompasses covering and prerequisite constraints. Then column generation is applied in order to improve the tractability of this large scale integer linear program. Here, the original problem is decomposed into a master and subproblem. The master problem is initialised with a set of dummy schedules to which we allocate the aircrew student population, whilst respecting class capacity limitations. The master problem then requests solutions from the subproblem that offer some promise of minimising the overall time spent in training. This process iterates between the master and subproblems until the solution of the master problem cannot be further improved and we have thus reached an optimal allocation of students to feasible schedules. Experimental results are compared with those of an ILP approach that assigns feasible schedules to labelled students.

Keywords: *Optimal timetabling, integer linear programming, network flow, column generation*

1 INTRODUCTION

This paper is concerned with the problem of optimally assigning trainee helicopter pilots in the Royal Australian Navy to a least-cost set of training schedules. The objective is to minimise the total time to graduate all students, as the total time to graduate is a proxy for both financial and morale costs. The key features of this problem are as follows. Students are required to pass a number of courses in order to graduate. Each course has multiple instances held at different times throughout the year and referred to as course sessions. Students must follow feasible schedules of course sessions, where a schedule is deemed feasible if it covers each course exactly once and satisfies certain prerequisite requirements. Some courses are run by external organisations. As such, TA-Avn does not have control over the times and capacities of all courses sessions. In (Bayliss et al., 2016), a stochastic tabu search approach is used to automate the timetabling process. However, in this paper, we adopt the technique of column generation as a first step toward the goal of obtaining an exact solution.

The approach adopted in this paper is to decompose the problem into two parts. The *subproblem* of designing a feasible schedule that covers all required courses and satisfies prerequisite constraints is formulated as a network flow problem. On the other hand, the *master problem* of optimally assigning students to feasible schedules whilst respecting course session capacity constraints is solved using column generation as a heuristic method. The idea is to iterate back and forth between the two problems. The solution of the master problem guides the search for an optimal solution of the subproblem and vice versa. The remainder of this paper is structured as follows. Section 2, which concerns the subproblem, provides a detailed explanation of the constraints to be satisfied by a feasible schedule. Section 3 examines the master problem. In Section 4 we compare the results of column generation to those obtained by an ILP. We conclude by offering suggestions for improvements to the current approach.

2 SUBPROBLEM FORMULATION

This section presents a set of constraints to be satisfied by a feasible schedule. The problem of designing a feasible schedule that minimises the value of some linear objective function is referred to as the column generation subproblem.

2.1 Representing the course prerequisite structure

The prerequisite structure can be represented by a digraph $G(V, A)$. Refer to Figure 1 for an illustration of the digraph associated with the prerequisites listed in Table 1. Let n_C denote the number of courses excluding the entry course. For $i = 0, \ldots, n_C$, the digraph $G(V, A)$ includes a vertex v_i corresponding to the ith course. For each pair (c_i, c_j) consisting of a course c_j and its prerequisite c_i (if it exists), the digraph $G(V, A)$ includes an arc a_{ij} pointing from vertex v_i to vertex v_j. Observe that the red arcs in Figure 1 may be omitted, because a flow along such arcs would bypass certain other vertices corresponding to prerequisite courses. In the example, course 2 is a prerequisite for courses 3 and 4. As course 3 is also a prerequisite for course 4, one cannot take course 4 immediately after taking course 2. Hence, a dashed arc is drawn between vertices v_2 and v_4. In general, an arc a_{ij} may be removed from the digraph $G(V, A)$ if its removal leaves a directed path from vertex v_i to vertex v_j. Introduce a sink v_F. If the ith course is not a prerequisite for any other course, then the vertex v_i is connected by an arc a_{iF} to the sink v_F. Finally, note that there is a partial order $<$ on the vertex set V such that $v_i < v_j$ if the vertex v_i precedes the vertex v_j in a directed path from the source v_0, corresponding to the entry course, to the sink v_F. If neither $v_i < v_j$ nor $v_j < v_i$, as is the case with the vertices v_4 and v_6 in Figure 1, then introduce the directed arcs a_{ij} and a_{ji}. Such extra arcs are illustrated in orange in Figure 1.

Table 1. Courses and prerequisites

Course	Prerequisite
1	0
2	1
3	2
4	2, 3
5	2, 4
6	2, 3

D. Kirszenblat et al, Using column generation to solve an aircrew training timetabling problem

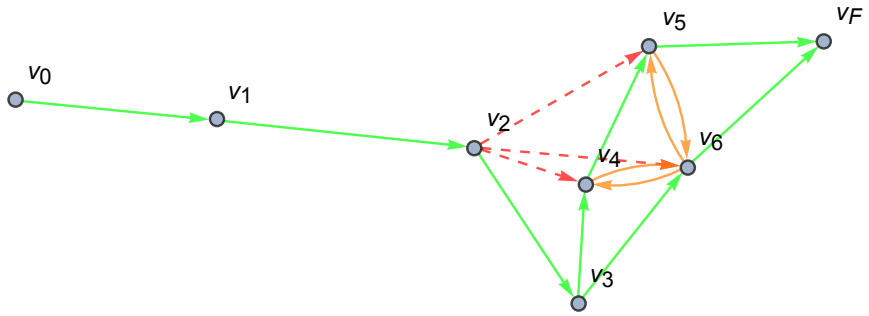


Figure 1. The digraph $G(V, A)$ represents the prerequisite structure

2.2 Exact cover constraints

Let c_{ij} denote the jth session of the ith course. (Refer to Table 2 for an example of course session input data.) Let z_{ij} denote a binary decision variable that is equal to 1 if course session c_{ij} is included in the schedule and 0 otherwise. The covering constraints ensure that each course is covered by exactly one session in a schedule.

$$\sum_{j} z_{ij} = 1 \qquad i = 0, 1, 2, \ldots, n_C \qquad \text{Exact cover} \tag{1}$$

Table 2. An example of course session input data

c_{ij}	Start Day	End Day	Capacity
c_{01}	53	53	9
c_{02}	115	115	15
c_{11}	77	217	30
c_{12}	140	279	18
c_{21}	84	226	27
c_{22}	182	325	26

2.3 Flow conservation constraints

Let y_{ij} denote a binary decision variable that is equal to 1 if the schedule flows directly from course i to course j and 0 otherwise. A feasible schedule must satisfy the following flow conservation constraints:

$$\sum_{a_{ij} \in A} y_{ij} = 1 \qquad i = 0, 1, \ldots, n_C \qquad \text{Unitary outflow} \tag{2}$$

$$\sum_{a_{ij} \in A} y_{ij} = 1 \qquad i = 1, 2, \ldots, F \qquad \text{Unitary inflow} \tag{3}$$

The first set of constraints says that the schedule flows out from every course. The second set of constraints says that the schedule flows into every course except the entry course and that the schedule terminates in exactly one course.

2.4 Prerequisite constraints

Let $\tau(i)$ denote the index set corresponding to the sessions of course i. Let S_{ij} denote the start day of course session c_{ij}. Similarly, let F_{ij} denote the end day of course session c_{ij} . Denote by $\mathcal{P}_i$ the set of prerequisites

for course i. We impose the following set of constraints:

$$\sum_{j\in\tau(i)} F_{ij} \times z_{ij} + 1 \leq \sum_{l\in\tau(k)} S_{kl} \times z_{kl} \qquad \text{for all } k \text{ and } i \in \mathcal{P}_k \tag{4}$$

This constraint ensures that the session of course i in the schedule must finish at least a day prior to the commencement of the session of course k in the schedule.

2.5 Time constraints

We require additional constraints to reflect the temporal order of course sessions. These constraints are similar in structure to the prerequisite constraints. Let ES_i denote the earliest start day of any session of course i. On the other hand, let LF_i denote the latest end day of any session of course i. Define the constant M_{ij} as the difference $LF_i - ES_j$. We define a *terminal arc* as an arc that is connected to the sink, and denote by $\tilde{A}$ the set of nonterminal arcs. For each nonterminal arc a_{ij} in $\tilde{A}$ such that M_{ij} is nonnegative, we impose the following constraint on the start and end times of its associated course sessions:

$$\sum_{j\in\tau(i)} F_{ij} \times z_{ij} + 1 - M_{ik}(1 - y_{ik}) \leq \sum_{l\in\tau(k)} S_{kl} \times z_{kl} \qquad \text{for all } a_{ik} \in \tilde{A} \text{ and } M_{ik} \geq 0 \tag{5}$$

To see how the time constraints work, suppose that the binary decision variable y_{ik} is turned on. That is, the schedule flows directly from course i to course k. Then the associated constraint ensures that session of course i included in the schedule ends at least one day prior to the commencement of the session of course k included in the schedule. If y_{ik} is turned off, then the associated constraint is inactive. Note that these constraints imply that a schedule corresponds to a Hamiltonian path through the vertices of the digraph $G(V, A)$. That is, subtours of the vertices of the digraph $G(V, A)$ cannot arise, as they do not permit a temporal ordering of course sessions.

2.6 Computing the makespan of a schedule

The *makespan* or duration of a schedule is a common measure of cost in scheduling problems. In order to compute the makespan of the schedule, we introduce a linear cost function. Let LS_0 denote the latest start time of any entry course session. (Note that there is only one entry course but possibly several entry course sessions.) Let t_0 denote the difference of the start time of the schedule and the latest start time LS_0 of any entry course session. Let t_F denote the end time of the schedule. Define the vector $\mathbf{v}$ to be $\mathbf{v} = (1, \mathbf{y}, \mathbf{z}, t_0, t_F)$. The first entry of the vector $\mathbf{v}$ is equal to 1 and the remaining entries are the decision variables of the subproblem. Define the vector $\mathbf{q}$ to be $\mathbf{q} = (LS_0, 0, 0, \ldots, -1, 1)$. That is,

$$\mathbf{q} \cdot \mathbf{v} = LS_0 - t_0 + t_F$$

Let τ_0 denote the index set corresponding to the entry course sessions. Let τ_F denote the index set corresponding to the terminal course sessions, i.e., the course sessions in which a schedule can terminate. We compute the makespan of the schedule by solving the problem to

$$\min \mathbf{q} \cdot \mathbf{v} \tag{6}$$

$$\text{subject to } t_0 \leq (S_{0j} - LS_0) z_{0j} \qquad \text{for all } c_{0j} \in \tau_0 \tag{7}$$

$$t_F \geq F_{ij} z_{ij} \qquad \text{for all } c_{ij} \in \tau_F \tag{8}$$

The constraints are defined so that the variable t_0 picks out the difference of the start time of the entry course session appearing in the schedule and the constant LS_0, whereas the variable t_F picks out the latest finish time of any terminal course session appearing in the schedule.

3 MASTER PROBLEM FORMULATION

The objective of the master problem is to assign a fixed number, say N, of students to a set of feasible schedules so that the total time spent in training is minimised. The key idea of column generation stems from the observation that the simplex method does not require all columns of the constraint matrix in order to find an optimal solution. Rather, columns that offer some promise of improving the value of the objective function can be generated as needed. In our master problem, the columns of the constraint matrix correspond to feasible

schedules. Column generation can therefore be used to identify feasible schedules that are likely to aid in finding an optimal solution as opposed to exhaustively enumerating all feasible schedules before optimising.

Suppose that there is a total of J feasible schedules satisfying the constraints of the subproblem. Once again, we emphasise that the feasible schedules do not need to be known in advance. For $j = 1, 2, \ldots, J$, let f_j denote the makespan of the jth feasible schedule. Let $\mathbf{f} = (f_1, f_2, \ldots, f_J)$ denote the cost vector. Similarly, for $j = 1, 2, \ldots, J$, let x_j denote the number of students assigned to the jth feasible schedule. Let $\mathbf{x} = (x_1, x_2, \ldots, x_J)$ denote the vector of integer decision variables. Let $\mathbf{z}_j$ denote the column of binary decision variables from the subproblem indicating which of the course sessions are included in the jth feasible schedule. The entries of the column $\mathbf{z}_j$ are treated as constants in the master problem. Similarly, let $\mathbf{b}$ denote the vector whose entries are the capacities of the respective course sessions. Now, the master problem is to

$$\min \quad \mathbf{f} \cdot \mathbf{x} \tag{9}$$

$$\text{subject to} \sum_{j=1}^{J} x_j = N \qquad \text{Student assignments} \tag{10}$$

$$\sum_{j=1}^{k} \mathbf{z}_j x_j \leq \mathbf{b} \qquad \text{Course session capacities} \tag{11}$$

$$\mathbf{x} \geq 0 \tag{12}$$

We choose to initialise the master problem using a single dummy column, interpreted as a dummy schedule to which we assign N students without violating the session capacity constraints. The dummy column need not be a feasible solution of the subproblem, provided we assign its master variable a sufficiently large cost. Assuming the master problem is feasible, when the algorithm terminates the dummy column will have either been driven out of the basis and replaced by some feasible solutions of the subproblem or its corresponding master variable will have been assigned the value of 0. We set the cost of the dummy schedule equal to M, where

$$M = \max\{LF_j : j = 1, 2, \ldots, n_c\} - \min\{ES_j : j = 1, 2, \ldots, n_c\} + 1$$

That is, M is the longest possible makespan of a schedule, and so the dummy schedule is at least as costly as any other schedule. Let $\mathbf{p}$ denote the vector of dual costs. (See, for example, (Bertsimas et al., 1997) for details of how to obtain the dual costs.) In order to identify a feasible schedule offering some promise of reducing the total time spent in training, we introduce the objective function $(\mathbf{q} - \mathbf{p})$ to the subproblem. That is, the subproblem is to

$$\min \quad (\mathbf{q} - \mathbf{p}) \cdot \mathbf{v} \tag{13}$$

subject to the exact cover, flow conservation, prerequisite and time constraints. The iterative method is carried out as follows. We turn our attention to the subproblem with the updated objective function. That is, we seek to minimise $(\mathbf{q} - \mathbf{p})$ over all schedules. If we find a schedule $\mathbf{v}_k$ for which $(\mathbf{q} - \mathbf{p}) \cdot \mathbf{v}_k$ is negative, then the corresponding column is said to have *negative reduced cost* and enters the basis. At the first iteration, at least one such schedule must exist if the subproblem is feasible. We then re-solve the master problem to find an optimal assignment of students to the updated set of schedules. The algorithm terminates when we are unable to find a column with negative reduced cost. At this stage we solve the master problem as an integer program in order to obtain an integral allocation of students to schedules. The reader is referred to [2] for a more detailed explanation of column generation.

Note that column generation is used as a heuristic method for solving this particular problem. That is, as many columns as needed are generated in order to solve the LP relaxation of the master problem, whereby fractional numbers of students are assigned to schedules. Then an integer solution is obtained by solving a restricted IP using only those schedules that have been obtained via column generation. As will be seen in the next section, column generation produces optimal solutions for the datasets that have been tested. However, in principle the optimal integer solution to the original problem and the optimal solution to the LP relaxation may not share the same set of variables, in which case the restricted master IP may produce a suboptimal answer or be infeasible. In the case where only the entry course sessions are at capacity, column generation will produce an optimal solution. To see this, note that for each entry course session, column generation will fill a shortest schedule with the required number of students. On the other hand, we have not obtained performance bounds for the

case where course sessions other than the entry course sessions are at capacity. However, an exact solution could be obtained by incorporating the column generation formulation presented here into a branch and price formulation.

4 EXPERIMENTAL RESULTS

In this section, we compare the results of column generation with those of an ILP according to which N labeled students are each assigned exactly one schedule. Both models were implemented using CPLEX and runs were performed on the same computer (Intel Core i5 processor with 8GB RAM) for comparison of execution times. The ILP model was implemented in Java, whereas column generation was implemented in MATLAB. As can be seen from the data in Table 3, column generation obtains the same results as the ILP, albeit more quickly for larger datasets. The respective values of the objective function are identical for both models and are not included in the table. Note that the last four rows of Table 3 pertain to data sets for which only the start and end times of intermediate course sessions were altered; the start and end times of entry and terminal course sessions were fixed. The difference in computational efficiency may be in part attributed to the high degree of symmetry associated with the ILP. There are two ways in which symmetry slows down the performance of the ILP. First, many optimal combinations of schedules may be identical after permuting the labels associated with the students. Second, an optimal combination of schedules may involve assigning the same schedule to different students. And as the number of students increases, so does the number of times that the same schedule has to be computed. This is very wasteful given that the problem is NP hard and the computation time is expected to grow exponentially with the size of the problem. On the other hand, column generation works in reverse by assigning a number of unlabelled students to schedules and is therefore insensitive to changes in the number of students.

Table 3. Experimental results

# Courses	# Sessions	ILP time (s)	CG time (s)
5	62	2.92	1.16
5	86	3.62	1.08
6	77	0.37	0.77
20	166	0.27	1.46
20	167	0.26	1.72
20	166	0.28	1.47
20	168	0.27	1.37
20	162	1.55	1.93
20	168	1.95	1.88
20	168	0.26	0.95
20	168	1.83	1.80
20	168	1.97	1.62

5 FUTURE DIRECTIONS

In future work we will modify the network flow formulation to account for pass rates associated with the courses and will implement branch and price in order to ensure that an exact solution is obtained. Another idea worth exploring is the possibility of warm starting the ILP solver by providing it with feasible solutions obtained by some other fast method.

6 CONCLUSIONS

This paper presents a column generation approach to optimally assigning students to training schedules. A comparison in terms of speed is made between the column generation approach and an ILP approach involving the assignment of feasible schedules to labelled students. Experimental evidence and symmetry arguments suggest that the column generation approach performs significantly faster as the size of the problem increases.

D. Kirszenblat et al, Using column generation to solve an aircrew training timetabling problem

REFERENCES

Bayliss, C., A. Novak, A. Nguyen, A. Moran, A. Caelli, S. Harrison, and S. Tracey (2016). Optimising aircrew training schedules using tabu search. *Australian Simulation Congress (SimTecT).*

Bertsimas, D., and J. N. Tsitsiklis (1997). Introduction to Linear Optimization. Athena Scientific, Massachusetts

1.3 Some modifications

Some material was omitted from the MODSIM paper and since submitting it, we have continued to extend the column generation algorithm, so that it now applies not only to the dummy datasets but to the real world problems faced by Defence. Several modifications are required. First, in practice, a student may not be able to complete all courses within the planning window. To accommodate this fact, we replace some of the exact cover constraints by inexact cover constraints and modify several of the other constraints accordingly. Second, some students may have already completed some courses when the planning window starts. Accordingly, we differentiate a number of "sub-syllabi", each of which includes the remaining courses in the syllabus to be followed by a particular group of students. As a result, the master problem now draws from a large number of subproblems as opposed to a single subproblem. Third, the session capacity constraints are now imposed on the expected numbers of students to be found in each of the sessions, assuming that each course is failed by a certain percentage of students on average. Finally, to guarantee that an integer solution is obtained, we are working on a branch & price algorithm based on the column generation algorithm [15].

1.3.1 Modified cover constraints

Let c_{ij} denote the jth session of the ith course. Let z_{ij} denote a binary decision variable that is equal to 1 if course session c_{ij} is included in the schedule and 0 otherwise. Let $\bar{z}_i$ denote a binary decision variable that is equal to 1 if course i is excluded from the schedule and 0 otherwise. The inexact covering constraints ensure that each course is covered by at most one session, except in the case of the entry course, which must be covered in a schedule:

$$\sum_j z_{0j} = 1 \qquad \text{(Exact cover)} \tag{1.1}$$

$$\sum_j z_{ij} + \bar{z}_i = 1 \qquad i = 1, 2, \ldots, n_C \qquad \text{(Inexact cover)} \tag{1.2}$$

1.3.2 Modified flow constraints

Let y_{ij} denote a binary decision variable that is equal to 1 if the schedule flows directly from course i to course j and 0 otherwise. A feasible schedule must satisfy the following flow constraints:

$$\sum_{a_{0j} \in A} y_{0j} = 1 \qquad \text{(Unitary outflow)}, \tag{1.3}$$

$$\sum_{a_{ij} \in A} y_{ij} = 1 + \bar{z}_i \qquad i = 1, \ldots, n_C \qquad \text{(Outflow)}, \tag{1.4}$$

$$\sum_{a_{ij} \in A} y_{ij} = 1 + \bar{z}_j \qquad j = 1, 2, \ldots, n_C \qquad \text{(Inflow)}. \tag{1.5}$$

$$\sum_{a_{iF} \in A} y_{iF} = 1 \qquad \text{(Unitary inflow)}. \tag{1.6}$$

The first constraint says that a schedule flows out from the entry course. The second set of constraints says that the schedule flows out from every course that it includes. The third set of constraints says that the schedule flows into every course that it includes, except the entry course, and that the schedule terminates in exactly one course.

1.3.3 Prerequisite constraints

Let $\tau(i)$ denote the index set corresponding to the sessions of course i. Let S_{ij} denote the start day of course session c_{ij}. Similarly, let F_{ij} denote the end day of course session c_{ij} . For any prerequisite pair (c_i, c_k) corresponding to a course k and its prerequisite course i, we impose the following constraint:

$$\sum_{j \in \tau(0)} F_{0j} \times z_{0j} + 1 + \leq \sum_{l \in \tau(k)} S_{kl} \times z_{kl} + (1 + M)\bar{z}_k, \qquad (c_0, c_k) \text{ a prerequisite pair.} \tag{1.7}$$

$$\sum_{j \in \tau(i)} F_{ij} \times z_{ij} + 1 + M\bar{z}_i \leq \sum_{l \in \tau(k)} S_{kl} \times z_{kl} + (1 + M)\bar{z}_k, \qquad (c_i, c_k) \text{ a prerequisite pair.} \tag{1.8}$$

This constraint ensures that the session of course i in the schedule must finish at least a day prior to the commencement of the session of course k in the schedule, assuming that both courses i and k are included in the schedule. If course i is included but course k is excluded from the schedule, then the constraint is inactive. If course i is excluded but course k is included in the schedule, then the constraint is violated. If both course i and course k are excluded from the schedule, then the constraint is trivially satisfied.

1.3.4 Time constraints

We require additional constraints to reflect the temporal order of course sessions. These constraints are similar in structure to the prerequisite constraints. Let ES_i denote the earliest start day of any session of course i. On the other hand, let LF_i denote the latest end day of any session of course i. Define the constant M_{ij} as the difference $LF_i - ES_j$. We define a *terminal arc* as an arc that is connected to the sink. With the exception of the terminal arcs, for each arc a_{ij} in A such that M_{ij} is nonnegative we impose the following constraint on the start and end times of its associated course sessions:

$$\sum_{j\in\tau(i)} F_{ij} \times z_{ij} + 1 - M_{ik}(1 - y_{ik}) \leq \sum_{l\in\tau(k)} S_{kl} \times z_{kl}, \qquad a_{ik} \text{ a nonterminal arc, } M_{ik} \geq 0. \quad (1.9)$$

To see how the time constraints work, suppose that the binary decision variable y_{ik} is turned on. That is, the schedule flows directly from course i to course k. Then the associated constraint ensures that session of course i included in the schedule ends at least one day prior to the commencement of the session of course k included in the schedule. If y_{ik} is turned off, then the associated constraint is inactive.

1.3.5 Pass rate constraints

We introduce a set of constraints in order to account for the pass rates associated with the courses. Let p_i denote the pass rate for course i, which is the same for all its sessions. With respect to a given schedule, let u_{ij} denote the cumulative pass rate for course session c_{ij}. That is, u_{ij} represents the expected number of students to enter course session c_{ij} when following a given schedule. For the entry course sessions, we have $u_{0j} = 1$ if the entry course session c_{0j} is included in the schedule and 0 otherwise. For the remaining course sessions, we have $u_{kl} = p_i \times u_{ij}$ if the course sessions c_{ij} and c_{kl} are included in the schedule. On the other hand, we have $u_{ij} = 0$ if the course session c_{ij} is not included in the schedule. To enforce these conditions, we impose the following set of constraints:

$$0 \leq u_{ij} \leq x_{ij} \qquad \text{(activation constraints)} \quad (1.10)$$

$$\sum u_{0j} = 1 \qquad \text{(entry course),} \quad (1.11)$$

$$-(1 - y_{ik}) + p_i \sum_{j\in\tau(i)} u_{ij} \leq \sum_{l\in\tau(k)} u_{kl} \leq p_i \sum_{j\in\tau(i)} u_{ij} + (1 - y_{ik}), \qquad e_{ik} \text{ a nonterminal edge.} \quad (1.12)$$

Note that if v_i is a cut-vertex in the digraph $G(V, A)$, then there is a simpler expression for its corresponding constraint. For a cut-vertex v_i, let P_i denote its set of predecessors. Then

$$\sum_{j \in \tau(i)} u_{ij} = \prod_{c_j \in P_i} p_j. \tag{1.13}$$

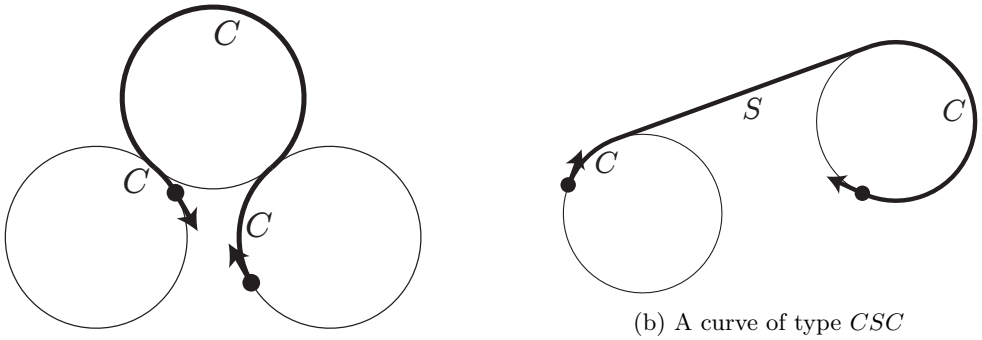
We now replace the column of binary decision variables $\mathbf{z}_j$ in the master problem formulation by the column of real-valued decision variables $\mathbf{u}_j$.

Chapter 2

A geometric approach to shortest curvature-constrained paths

2.1 Introduction

This chapter concerns the following problem about shortest curvature-constrained paths. Suppose that you wish to drive a vehicle along a shortest path between two points A and B in the plane. You depart from the point A in a certain direction given by a tangent vector $\vec{A}$ and are required to arrive at the point B from a certain direction given by a tangent vector $\vec{B}$. Ordinarily, the shortest path between the points A and B would be a straight-line segment. However, the vehicle has a minimum turning radius and hence the path must be curvature-constrained. This problem was first studied by Markov in 1887 [37]. Later, in 1957, Dubins proved that all length minimising curvature-constrained curves are concatenations of circular arcs and straight-line segments, referred to as *components* [20]. Let C denote "circular arc" and S denote "straight-line segment". Each shortest curvature-constrained path can be designated by a sequence of C and S symbols, depending on how it is composed of circular arcs and straight-line segments. Such a sequence is called a *type*. For example, Fig. 2.1 depicts two length minimising curves, one of type CCC and one of type CSC. Dubins showed that each length minimising curve is necessarily of type CCC or CSC, or a subtype thereof (such as CC.) Problems of this nature are typically tackled from the viewpoint of optimal control theory (see, e.g., [28]). However, in our paper, we derive Dubins' results using more direct geometric arguments, which afford additional insights into the nature of the problem and allow us to extend the results to different settings, e.g., ambient spaces of positive or negative curvature. A useful physical model to keep in mind is one comprising an elastic band or string and pulleys. The string, which represents the path, is pulled taut, so as to minimise length, and the pulleys serve to enforce the curvature constraint.



(a) A curve of type CCC

(b) A curve of type CSC

Figure 2.1: Dubins curves.

Two important applications of curvature-constrained paths are in path-planning for unmanned aerial vehicles or drones and in the network design of underground mines, as will be discussed in the next chapter.

The general strategy that is adopted in the paper is a sort of perturbational argument. That is, we take a path that does not fit Dubins' classification of length minimising curves and show that we can reduce its length by modifying it. In particular, we take an arbitrary curvature-constrained path and homotope it to a shorter path consisting of constant curvature segments, i.e., arcs of circles and straight-line segments. In this context, we use the term *complexity* to refer to the number of components of a path, i.e., the number of arcs of circles and straight-line segments. The next step is to reduce the *complexity* of the path by replacing certain subpaths with other subpaths. Finally, we reduce the analysis to the study of paths comprising no more than 4 arcs of circles or straight-line segments. The last case of a complexity 4 path to be tackled is that of a path of type $CCCC$, see Proposition 3.7 and the configuration of four Dubins circles in Fig. 8 in the paper. This is a crucial yet relatively small step in the argument, and seeing as the author contributed less than 50 per cent of the work to the paper, according to the University of Melbourne guidelines, we present in this chapter only the material from the paper that is relevant to Proposition 3.7, whereas the entire paper can be found in the appendix. For the case of a path of type $CCCC$, we use a variational approach. We study the derivative of the length of the path as a function of the configuration of circular arcs and assert that if the first derivative is nonzero or the second derivative negative, which is the case for a path of type $CCCC$, then we can perturb the configuration to obtain a shorter path.

2.2 A geometric approach to shortest bounded curvature paths

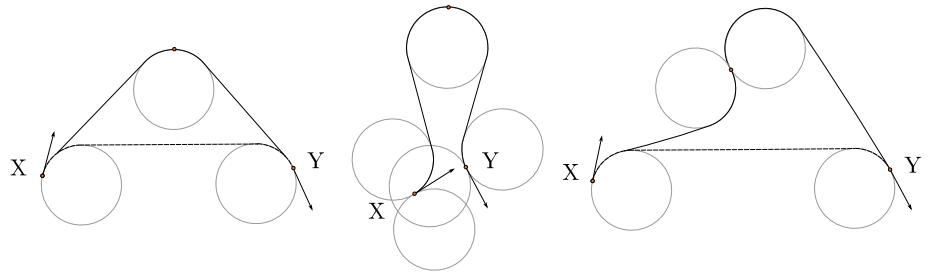


FIGURE 7. Examples of components of type $\mathcal{C}_1$ and $\mathcal{C}_2$. The dashed traces in the left and right figures are replacements. The middle illustration corresponds to a non-admissible component of type $\mathcal{C}_1$. Note that no replacement can be constructed for such a component.

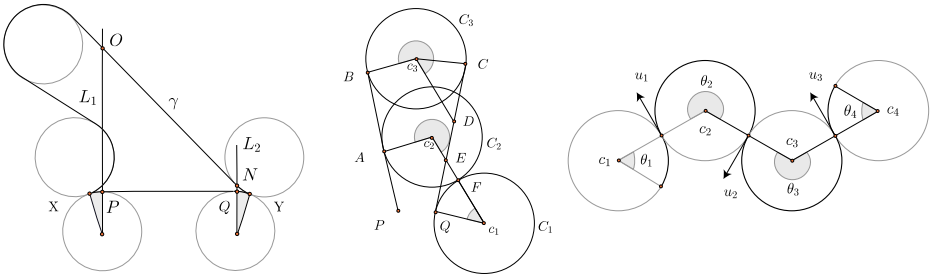


FIGURE 8. Left: The restriction of a *cs* path γ to an admissible component of type $\mathcal{C}_1$. Lemma 2.14 is trivially adapted to a component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$) (see notation in Figure 6). Center: The notation in Proposition 3.4. The path γ is the SCS path from P to Q. Right: The notation for the non optimality of CCCC paths.

Theorem 3.4. *Components of type $\mathcal{C}_1$ and $\mathcal{C}_2$ are not length minimisers.*

Proof. Consider a component of type $\mathcal{C}_1$. If the component is admissible then by applying Proposition 3.3 the result follows. If the component of type $\mathcal{C}_1$ is non-admissible then consider the SCS part of the component of type $\mathcal{C}_1$, denote it by γ, and without loss of generality suppose the line segments in γ have the same length. Referring to Figure 8 (center) for notation, we proceed to construct a path shorter than γ. Note that γ is the SCS path from P to Q with the length of $\overline{PB}$ and $\overline{CQ}$ being the same. Denote by C_1 the left adjacent circle at Q. Let C_2 be the circle that is simultaneously tangent to C_1 and $\overline{PB}$. Let C_3 be the circle containing the middle component of γ; denote the centers of these circles in lowercase. The parallel line to $\overline{c_1\,c_2}$ passing through c_3 intersects $\overline{CQ}$ at D. The segment $\overline{c_1\,c_2}$ intersects $\overline{CQ}$ at E. Since C_3 is above C_2 (both circles being tangent to $\overline{PB}$) then by applying Lemma 2.8 we conclude that the length of γ between the points B and D is greater than the length of the arc in C_2 between A and F. Again by Lemma 2.8 we conclude that the length of the shorter arc between Q and F in C_1 is less

than the length of the segment $\overline{QE}$. Denote by δ the SCC path from P to Q with first component $\overline{PA}$, second component the arc AF in C_2 and third component the arc FQ lying in C_1. We conclude that $\mathcal{L}(\delta) < \mathcal{L}(\gamma)$ implying that the component of type $\mathcal{C}_1$ is not a length minimiser. If a component of type $\mathcal{C}_2$ is admissible then by applying Lemma 2.8 the result follows. If a component of type $\mathcal{C}_2$ is non-admissible, then by applying an identical argument as in the previous paragraphs the result follows. □

Corollary 3.5. The SCS and CCS (or SCC) paths are not length minimisers.

Proof. By the construction in Theorem 3.4 we have that SCS paths are not length minimisers. The proof that CCS (or SCC) paths are also not length minimisers follows from an identical construction as the one used for the non-admissible case in Theorem 3.4. We leave the details to the reader. □

Proposition 3.6. A CCC path with middle arc of length less than or equal to π is not a length minimiser.

Proof. Without loss of generality consider an LRL path. An LRL path with middle component of length less than or equal to π has the center of the circle containing R below the line joining $c_l(\mathrm{X})$ and $c_l(\mathrm{Y})$. Consider the adjacent circles $\mathrm{C}_l(\mathrm{X})$ and $\mathrm{C}_l(\mathrm{Y})$ and construct a replacement as in Proposition 2.13. Apply Lemma 2.14 to the RLR path to conclude the statement. If we consider an RLR path with middle component of length less than or equal to π then consider $\mathrm{C}_r(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{Y})$ and proceed as before. □

Proposition 3.7. The *cs* paths of complexity four are not length minimisers.

Proof. If a complexity four path contains exactly one line segment then it must contain a CCS or SCC component. By virtue of Corollary 3.5 we have that such a component is not a length minimiser. If a complexity four path contains exactly two line segments then it must contain an SCS component. Again by Corollary 3.5 we have that such a component is not a length minimiser.

For a proof that CCCC paths are not length minimisers, we use a variational argument. In particular, we show that if a CCCC path is a critical point of the length function, then it is unstable. So that it is evident how the equations scale as the minimum turning radius κ varies, we do not set κ equal to unity. Let γ be a CCCC path whose arcs have lengths $\theta_1(t), \theta_2(t), \theta_3(t)$ and $\theta_4(t)$ and centres $c_1(t), c_2(t), c_3(t)$ and $c_4(t)$, respectively. For succinctness, we suppress the dependency on t. Hence the length of γ is given by

$$\mathcal{L}(\gamma) = \theta_1 + \theta_2 + \theta_3 + \theta_4.$$

Consider a perturbation of γ to a nearby path of type CCCC. The first variation of length is given by

$$\dot{\mathcal{L}}(\gamma) = \langle (c_2 - c_1)', \hat{u}_1 \rangle + \langle (c_3 - c_2)', \hat{u}_2 \rangle + \langle (c_4 - c_3)', \hat{u}_3 \rangle,$$

where the unit vectors $\hat{u}_i$ are tangent to the circular arcs of γ with centres c_i and c_{i+1} and have the same orientation as γ (see Figure 8 right). Note that the lengths $\|c_{i+1} - c_i\|$ remain constant throughout the perturbation. Hence, the vector $(c_{i+1} - c_i)'$ must be perpendicular to the vector $c_{i+1} - c_i$. In other words, the vector $(c_{i+1} - c_i)'$ must be parallel to the vector $\hat{u}_i$. So we obtain $\langle (c_{i+1} - c_i)', \hat{u}_i \rangle = \pm \|(c_{i+1} - c_i)'\|$, where the sign of the inner product depends on the direction of

variation. In particular, the inner products $\langle (c_2 - c_1)', \hat{u}_1 \rangle$ and $\langle (c_4 - c_3)', \hat{u}_3 \rangle$ are of opposite sign. On the other hand, the inner product $\langle (c_3 - c_2)', \hat{u}_2 \rangle$ may change sign relative to $\langle (c_2 - c_1)', \hat{u}_1 \rangle$ and $\langle (c_4 - c_3)', \hat{u}_3 \rangle$. Note that the centres c_1 and c_4 are fixed. Hence both $c_1' = 0$ and $c_4' = 0$. At the critical point, we set the derivative of the length function $\mathcal{L}(\gamma)$ equal to zero and obtain one of the following two equations:

$$\dot{\mathcal{L}}(\gamma) = \|c_2'\| + \|c_3' - c_2'\| - \|c_3'\| = 0,$$

or

$$\dot{\mathcal{L}}(\gamma) = \|c_2'\| - \|c_3' - c_2'\| - \|c_3'\| = 0.$$

In either case, the vectors c_2' and c_3' must be parallel by the triangle inequality. Since the vector $c_3 - c_2$ cannot undergo any change in length, the perturbation vectors c_2' and c_3' must in fact be equal. Note that the perturbation vectors c_2' and c_3' are tangent to the circles of radius 2κ with centres c_1 and c_4. It is geometrically obvious that a stationary configuration is obtained only in the symmetric situation where $\hat{u}_1 = \hat{u}_3$. In order to demonstrate that this configuration corresponds to an unstable critical point of the length function, we will need to study the second variation of length. By the Leibniz rule,

$$\ddot{\mathcal{L}}(\gamma) = \langle (c_2 - c_1)'', \hat{u}_1 \rangle + \langle (c_2 - c_1)', \hat{u}_1' \rangle + \langle (c_3 - c_2)'', \hat{u}_2 \rangle + \langle (c_3 - c_2)', \hat{u}_2' \rangle + \langle (c_4 - c_3)'', \hat{u}_3 \rangle + \langle (c_4 - c_3)', \hat{u}_3' \rangle.$$

By the same reasoning as above, the vector $(c_{i+1} - c_i)'$ must be perpendicular to the vector $\hat{u}_i'$, and both $c_1'' = 0$ and $c_4'' = 0$. Moreover, at the critical point, $\hat{u}_1 = \hat{u}_3$. Therefore,

$$\ddot{\mathcal{L}}(\gamma) = \langle (c_3 - c_2)'', \hat{u}_2 \rangle - \langle (c_3 - c_2)'', \hat{u}_1 \rangle.$$

The vector $(c_3 - c_2)''$ makes an obtuse angle with the vector $\hat{u}_2$ throughout the perturbation and is parallel to the vector $\hat{u}_2$ at the critical point. To see this, note that the component of $(c_3 - c_2)''$ parallel to $c_3 - c_2$ is equal to $-|\langle (c_3 - c_2)', \hat{u}_2 \rangle|^2 / 2\kappa$, and akin to the centripetal acceleration of the vector $c_3 - c_2$. At the critical point, $c_2' = c_3'$. Hence, the vector $c_3 - c_2$ is not subject to any rotation and the component of $(c_3 - c_2)''$ perpendicular to $c_3 - c_2$ must be zero. Therefore,

$$\ddot{\mathcal{L}}(\gamma) = -\|(c_3 - c_2)''\| - \langle (c_3 - c_2)'', \hat{u}_1 \rangle < 0,$$

and the configuration is unstable. □

Corollary 3.8. *cs* paths of complexity greater than 3 are not length minimisers.

Proof. Immediate from Proposition 3.7. □

Now we proceed to establish the classification of length minimisers in spaces of planar bounded curvature first obtained in [14].

Theorem 3.9. *Choose* $\mathrm{X}, \mathrm{Y} \in T\mathbb{R}^2$. *A length minimising bounded curvature path in* $\Gamma(\mathrm{X},\mathrm{Y})$ *is either a* CCC *path having its middle component of length greater than* π *or a* CSC *path. Some of the circular arcs or line segments can have zero length.*

Proof. Choose $\mathrm{X}, \mathrm{Y} \in T\mathbb{R}^2$ and consider a fragmentation for $\gamma \in \Gamma(\mathrm{X},\mathrm{Y})$. If γ is not a *cs* path then by applying Lemma 2.14 to each fragment we obtain a *cs* path in $\Gamma(\mathrm{X},\mathrm{Y})$ shorter than γ. We conclude that γ is not a length minimiser in $\Gamma(\mathrm{X},\mathrm{Y})$. If γ is a *cs* path of complexity greater than or equal to 4, by Corollary 3.8 we conclude that γ is not a length minimiser in $\Gamma(\mathrm{X},\mathrm{Y})$. If the complexity of γ is exactly 3 then by Corollary 3.5 we have that SCS and CCS (or SCC) paths are not length minimisers. By applying Proposition 3.6 we conclude the proof. □

Chapter 3

Minimal curvature-constrained networks

3.1 Introduction

This chapter concerns a novel combination of the Dubins' problem, as discussed in the previous chapter, and the Steiner tree problem, which we will describe now [21]. The earliest version of the Steiner tree problem dates back to around 1643, when Fermat posed the problem of finding the optimal location of a junction interconnecting three points, which can be interpreted as the locations of three cities [7]. The location of the junction is supposed to be optimal in the sense that the resulting road network is of minimum length. See Fig. 3.1 for an illustration. Torricelli provided an algorithm for the construction of the junction. The circles and straight-line segments of the algorithm are also illustrated in Fig. 3.1. Then, in 1961, Melzak generalised Torricelli's construction to provide a method for constructing minimal networks interconnecting an arbitrarily large number of points in the plane [41]. Such minimal networks can be considered as one-dimensional minimal surfaces and have been modelled as soap films and systems comprising strings and weights [42]. One example of an application of Melzak's algorithm is in the design of minimum cost communication networks [22]. Another area in which Steiner trees naturally arise is in the design of least cost networks for underground mining. In the following paper, we provide two algorithms for the design of minimal curvature-constrained networks with applications in the design of underground mines [30]. The curvature-constraint accounts for the minimum turning radius of a truck to be driven along a path of the network – typically around 50 m. The first algorithm is an analogue of Melzak's algorithm and can be used to construct a minimal curvature-constrained network interconnecting an arbitrarily large number of points in the plane. The second algorithm is a gradient descent algorithm for the construction of a

minimal-curvature-constrained network interconnecting three points in 3D-space.

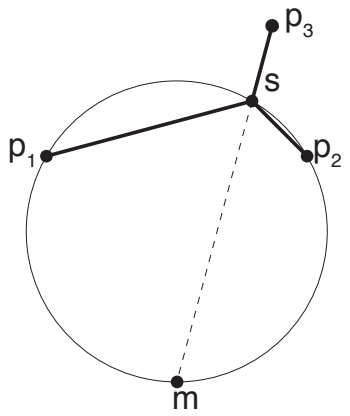


Figure 3.1: Torricelli's construction.

3.2 Minimal curvature-constrained networks

Noname manuscript No.
(will be inserted by the editor)

Minimal curvature-constrained networks

D. Kirszenblat · K.G. Sirinanda · M. Brazil · P.A. Grossman · J.H. Rubinstein · D.A. Thomas

Received: date / Accepted: date

Abstract This paper introduces an exact algorithm for the construction of a shortest curvature-constrained network interconnecting a given set of directed points in the plane and a gradient descent method for doing so in 3D space. Such a network will be referred to as a minimum Dubins tree, since its edges are Dubins paths (or slight variants thereof). The problem of constructing a minimum Dubins tree appears in the context of underground mining optimisation, where the objective is to construct a least-cost network of tunnels navigable by trucks with a minimum turning radius. The Dubins tree problem is similar to the Steiner tree problem, except the terminals are directed and there is a curvature constraint. We propose the minimum curvature-constrained Steiner point algorithm for determining the optimal location of the Steiner point in a 3-terminal network. We show that when two terminals are fixed and the third varied in the planar version of the problem, the Steiner point traces out a limaçon.

Keywords Network optimisation · Optimal mine design · Dubins path · Curvature constraint · Steiner point.

K.G. Sirinanda · P.A. Grossman · D.A. Thomas
Department of Mechanical Engineering, The University of Melbourne, Melbourne, Victoria 3010, Australia.
Tel.: +61 3 8344 6734
Fax: +61 3 8344 4290

M. Brazil
Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, Victoria 3010, Australia.
Tel.: +61 3 8344 6699
Fax: +61 3 8344 7412

D. Kirszenblat · J.H. Rubinstein
School of Mathematics and Statistics, The University of Melbourne, Melbourne, Victoria 3010, Australia.
Tel.: +61 3 8344 5550
Fax: +61 3 8344 4599
E-mail: dki@student.unimelb.edu.au

1 Introduction

This paper is concerned with the problem of designing a shortest path network for vehicles. The paths in the network are subject to a curvature constraint that accounts for the minimum turning radius of the vehicles. Interest in such networks is motivated by their relevance for designing the access in underground mines [3].

The Dubins tree problem can be viewed as a combination of two problems that have been well-studied in the optimisation literature: the Steiner problem and the Dubins problem. What follows is a review of these two problems with basic geometric and topological concepts being drawn primarily from [5] and [9].

First, consider Fermat's problem, which is a 3-terminal special case of the Steiner problem:

Problem 1 (Fermat's problem) Given three points p_1, p_2, p_3 in the plane $\mathbb{R}^2$, find the point s which minimises the sum of the distances $||sp_1|| + ||sp_2|| + ||sp_3||$.

The uniqueness of s, called the *Steiner point*, is obtained from the convexity of the Euclidean norm. If one of the angles of $\triangle p_1p_2p_3$ is at least 120°, then s is located at its vertex. Otherwise, s lies in the interior of $\triangle p_1p_2p_3$, whose sides subtend angles of 120° at s. Melzak [10] proposed a ruler and compass construction for finding s in the latter case. Let m be the third vertex of the equilateral triangle with p_1 and p_2 as its other two vertices, and whose interior lies outside that of $\triangle p_1p_2p_3$. Let Γ be the circle through p_1, p_2, m. Then s is the intersection of Γ and the *Simpson line* mp_3 as shown in Fig. 1.

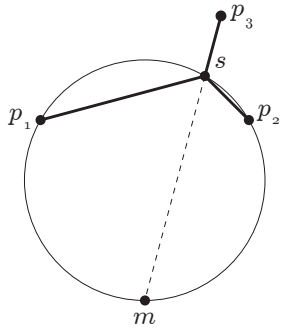


Fig. 1 Melzak's construction.

Based on the above 3-terminal algorithm that substitutes m for p_1 and p_2, locates s and adds the straight segments p_1s and p_2s, the Melzak algorithm generalises to n terminals (see [10] for details). The Melzak algorithm yields all the minimising networks of the Steiner problem:

Problem 2 (The Steiner problem) Given n points $p_1, p_2, \ldots, p_n$ in the plane $\mathbb{R}^2$, construct a shortest network interconnecting these n points.

We now define the Dubins problem in the plane. It is convenient to introduce some notation. A directed point is a pair $(p, \vec{p})$, where p is a point in the plane

$\mathbb{R}^2$ and $\vec{p}$ is a tangent vector in the tangent space $T_p\mathbb{R}^2$ based at p. For simplicity, such a pair is denoted as $\mathbf{p}$. Given two directed points $\mathbf{p}_1, \mathbf{p}_2$, a path connecting these two directed points is called *admissible* [1] if:

(i) It is continuously differentiable.
(ii) It starts at p_1 and finishes at p_2 and has tangent vectors $\vec{p}_1$ and $\vec{p}_2$ at these two points, respectively.
(iii) It is piecewise twice differentiable; there is a finite number of points at which the curvature is not defined.
(iv) Where the curvature exists it is less than or equal to a fixed positive number ρ^{-1}, where ρ is the minimum radius of curvature.

The Dubins problem can now be formulated as follows:

Problem 3 (The Dubins problem) Given an initial directed point $\mathbf{p}_1$ and a final directed point $\mathbf{p}_2$, construct a shortest admissible path connecting these two directed points.

Dubins [5] proved that such a path, called a *Dubins path*, necessarily exists and consists of not more than three components, each of which is either a straight segment, denoted by S, or an arc of a circle of radius ρ, denoted by C. Furthermore, such a path is necessarily a subpath of a path of type CSC or of type CCC.

We now formulate the network version of the problem. Given three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ in the plane, a network interconnecting these three directed points is called *admissible* if:

(i) It is continuously differentiable except at any Steiner point.
(ii) It interconnects p_1, p_2, p_3 and has tangent vectors $\vec{p}_1, \vec{p}_2, \vec{p}_3$ at these three points, respectively, when traveling toward the Steiner point.
(iii) It is piecewise twice differentiable; there is a finite number of points at which the curvature is not defined.
(iv) Where the curvature exists it is less than or equal to a fixed positive number ρ^{-1}, where ρ is the minimum radius of curvature.

In section 2, we present a generalisation of the Melzak algorithm for solving the following problem:

Problem 4 (The 3-terminal Dubins tree problem in the plane) Given three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, construct a shortest admissible network interconnecting these three directed points.

An example of such a network is shown in Fig. 2. By analogy with Steiner trees, a local minimum network is called a *Dubins tree*, whereas a global minimum network is called a *minimum Dubins tree*.

Finally, we define the problem of constructing a 3-terminal Dubins tree in 3D space. As will be explained, such a network can be viewed as a lifted version of a planar 3-terminal Dubins tree. From an engineering standpoint, i.e. for the purpose of construction and navigability, a Dubins tree in 3D space should have the following two properties. First, each edge of the network should be of constant gradient. Note that the constant gradient of one edge is allowed to differ from the constant gradient of another edge. In practice, an upper bound of $\frac{1}{7}$ on the gradient of each edge is required. We will not, however, incorporate the upper

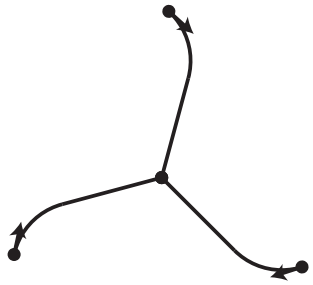


Fig. 2 A 3-terminal Dubins tree in the plane.

bound on the gradient in this paper. Second, there should be an upper bound on the curvature of the horizontal projections of all edges of the network. In other words, the horizontal projections of all edges of the network should be admissible paths, i.e., have curvature less than or equal to ρ^{-1} wherever it is defined. Together these two conditions imply that the arcs of a Dubins tree in 3D space must be helical arcs. To see this, observe that the horizontal projection of a shortest path in 3D space that satisfies the curvature constraint is a Dubins path in the plane. Moreover, an arc projecting to the arc of a circle lies on a cylinder. The constant gradient constraint then implies that the arc is helical. See Fig. 3 for an illustration of a minimum Dubins tree in 3D space.

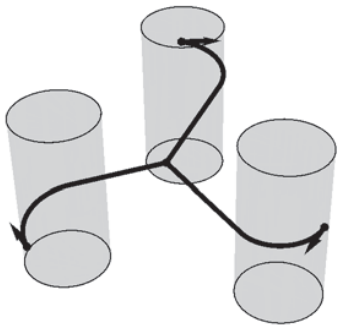


Fig. 3 A 3-terminal Dubins tree in 3D space.

To state the problem more precisely, we define a directed point in 3D space as a pair $(p, \vec{p})$, where p is a point in $\mathbb{R}^3$ and $\vec{p}$ is a tangent vector in the horizontal tangent space $T_p\mathbb{R}^2$ based at p. Such a pair is denoted as $\mathbf{p}$. Given three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ in 3D space, a network interconnecting these three directed points is called *admissible* if:

(i) It is continuously differentiable except at any Steiner point.
(ii) It interconnects p_1, p_2, p_3 and has tangent vectors with horizontal components given by $\vec{p}_1, \vec{p}_2, \vec{p}_3$ at these three points, respectively, when traveling toward the Steiner point.

(iii) It is piecewise twice differentiable; there is a finite number of points at which the curvature is not defined.
(iv) Where the horizontal projection of the curvature exists it is less than or equal to a fixed positive number ρ^{-1}, where ρ is the minimum radius of curvature.
(v) Each edge e_1, e_2, e_3 of the network is of constant gradient g_1, g_2, g_3.

In section 3, we present the minimum curvature-constrained Steiner point algorithm for numerically approximating the solution of the following problem:

Problem 5 (The 3-terminal Dubins tree problem in 3D space) Given three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ in 3D space, construct a shortest admissible network interconnecting these three directed points.

2 Minimal curvature-constrained networks in the plane

In this section, we provide a ruler and compass algorithm for constructing Dubins trees in the plane. The ruler and compass algorithm is analogous to the Melzak algorithm for solving Fermat's problem, which was illustrated in the introduction. We note that Weng provides a generalisation of the Melzak algorithm for constructing Dubins trees in the plane in [14]. We extend Weng's algorithm by allowing for the assignment of costs or weights to the edges of the network. The rationale behind such a generalisation is as follows. Suppose that one wishes to draw a distinction between major and minor roads of a network and to shorten the former at the expense of lengthening the latter. Such an outcome can be achieved by assigning proportionally larger costs of construction and haulage or weights to the major roads of the network.

In what follows, we will restrict our attention to full Dubins trees, whose definition can perhaps be most readily grasped with reference to their analogues, full Steiner trees. Recall that in Fermat's problem, if all the angles of $\triangle p_1p_2p_3$ are less than 120°, then the network consists of three edges meeting at a Steiner point. Such a network is called a *full Steiner tree*. On the other hand, if one of the angles of $\triangle p_1p_2p_3$ is no less than 120°, then the Steiner point coincides with its vertex and the network is comprised of only two edges. Such a network is referred to as a *degenerate Steiner tree*. Moreover, if one of the angles of $\triangle p_1p_2p_3$ is greater than 120°, then the Melzak algorithm fails to produce a Steiner point. In this sense, degenerate networks can be seen as (limiting) cases in which the Melzak algorithm fails. In the Dubins tree problem, a similar failure of the generalised Melzak algorithm to construct a Steiner point can occur. The conditions under which the arrangement of terminals does not give rise to a full Dubins tree are messier than in the analogous case of Fermat's problem and are beyond the scope of this paper. For simplicity, we will view a full Dubins tree on three terminals as a tree, each edge of which is of type CS when traveling from the terminal toward the Steiner point. Literature search suggests that the degenerate cases have not been studied.

The first task is to determine the angles at the Steiner point between the straight segments of the network. We will use a variational argument, inspired by the work of Rubinstein and Thomas on Steiner trees [12], that relies on the following proposition.

Proposition 1 *Let pqs denote a curve of type CS and suppose that the point p is held fixed while the point s is varied. Then the first variation of length of the*

curve pqs is the scalar product between the direction of variation of s and the unit vector $\hat{qs}$.

Proof Let *pqs* be a curve of type *CS* where:

- *pq* is a circular arc of radius ρ.
- *qs* is a straight segment.

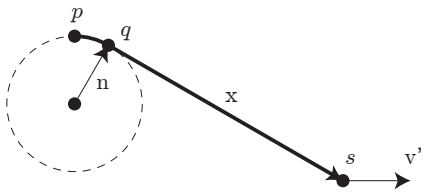


Fig. 4 A curve of type *CS*.

In Fig. 4, the point s moves along any smooth curve with derivative at its initial position being the vector $\mathbf{v}'$. Let L denote the length of the curve *pqs*. The curve *pq* is an arc of a circle of radius ρ centred at the origin. The length of *pq* is given by the product of the radius ρ and the angle θ subtended by the arc *pq* at the origin. Let $\mathbf{n}$ denote the vector from the origin to the point q. Let $\mathbf{x}$ denote the vector from the point q to the point s with magnitude x and direction given by the unit vector $\hat{\mathbf{x}}$. Observe that the first variation of the vector $\mathbf{x}$ is equal to the first variation of its head, i.e. $\mathbf{v}'$, minus the first variation of its tail, i.e. $\mathbf{n}'$. If the point s is perturbed in the direction of $\mathbf{v}'$, then the first variation of length of the curve *pqs* is

$$\begin{aligned} L' &= \rho\theta' + x' \\ &= \mathbf{n}' \cdot \hat{\mathbf{x}} + \mathbf{x}' \cdot \hat{\mathbf{x}} \\ &= \mathbf{n}' \cdot \hat{\mathbf{x}} + (\mathbf{v}' - \mathbf{n}') \cdot \hat{\mathbf{x}} \\ &= \mathbf{v}' \cdot \hat{\mathbf{x}}. \end{aligned}$$

□

It is instructive to consider the extreme case of the Dubins tree problem in which the minimum turning radius is zero. If the edges of the Dubins tree are unweighted, then we recover the classic Steiner tree problem. On the other hand, if the edges of the Dubins tree are assigned weights, then we recover the Fermat-Weber problem [6]. In this extreme case, we can easily determine the angles between the straight segments of the network. Let us focus on the problem of determining the angles between the straight segments of an unweighted 3-terminal network. For $i = 1, 2, 3$, let $\mathbf{x}_i$ denote the vector pointing from the ith terminal to the variable Steiner point. For $i = 1, 2, 3$, let w_i denote the weight assigned to the ith path of the network, i.e., the cost per unit length of the path. The objective is to minimise the weighted sum $\sum_{i=1}^{3} w_i \|\mathbf{x}_i\|$. We use a variational argument that involves

varying the position of the Steiner point along any smooth curve with derivative at its initial position being the vector $\mathbf{v}'$. Noting that for $i = 1, 2, 3$, $\mathbf{x}_i' = \mathbf{v}'$ and $\mathbf{x}_i \cdot \hat{\mathbf{x}}_i' = 0$ throughout the perturbation, we have by the product rule,

$$\begin{aligned} L' &= \sum_{i=1}^{3} w_i \|\mathbf{x}_i\|' \\ &= \sum_{i=1}^{3} w_i (\mathbf{x}_i \cdot \hat{\mathbf{x}}_i)' \\ &= \sum_{i=1}^{3} w_i (\mathbf{x}_i' \cdot \hat{\mathbf{x}}_i + \mathbf{x}_i \cdot \hat{\mathbf{x}}_i') \\ &= \sum_{i=1}^{3} w_i \mathbf{v}' \cdot \hat{\mathbf{x}}_i \\ &= \mathbf{v}' \cdot \sum_{i=1}^{3} w_i \hat{\mathbf{x}}_i. \end{aligned}$$

At equilibrium we must have $\sum_{i=1}^{3} w_i \hat{\mathbf{x}}_i = 0$. There is a mechanical interpretation of the equilibrium condition. Suppose that three tensions f_1, f_2 and f_3, proportional to the three weights w_1, w_2 and w_3, respectively, pull along the edges of the network and away from the Steiner point. The equilibrium condition says that the three tensions must cancel each other out. An application of the law of cosines gives the angles α_i at the Steiner point in terms of the weights w_i with $i = 1, 2, 3$.

Proposition 2 *For any 3-terminal Dubins tree whose paths are of type CS, the three Dubins paths meet at angles $\alpha_1, \alpha_2, \alpha_3$ determined as in the case of the Fermat-Weber problem.*

Fig. 5 A perturbation in the direction of negative gradient will shorten the network.

Proof It suffices to work locally. Fix a small disk around the Steiner point, see Fig. 5 (left). Fix the points of intersection of the Dubins paths with the boundary of the disk. Within the disk, if the three straight segments do not make angles $\alpha_1, \alpha_2, \alpha_3$ with each other, then move the Steiner point in the direction of negative gradient, see Fig. 5 (right). Recall that the gradient is the sum of the three weighted unit

vectors pointing toward the Steiner point s. Finally, replace the newly created angles at the boundary of the disk by small arcs of circles of radius ρ. Note that two length-reducing moves have been made. This shows that the three straight segments meet at angles $\alpha_1, \alpha_2, \alpha_3$. □

We now present a generalisation of the Melzak algorithm for constructing a weighted minimum Dubins tree interconnecting three given directed points in the plane. The lengths corresponding to the three weights w_1, w_2, w_3 might not be constructible by ruler and compass. However, supposing that we can mark off lengths corresponding to the three weights w_1, w_2, w_3, then all subsequent constructions, i.e. circles and lines, can be performed using the ruler and compass in the sense of Euclid.

We first note that a directed point $\mathbf{p} = (p, \vec{p})$ is associated with two Dubins circles, both of which are incident to the point p and tangent to the tangent vector $\vec{p}$ based at p. The C component of a Dubins path of type CS from a terminal to a Steiner point can, depending on the data of the problem, be oriented clockwise or anticlockwise. When attempting to construct a minimum Dubins tree, we should then consider all possible combinations of orientations of C components. Given that we are working with three directed points, we need to consider a total of eight combinations. Each combination can be considered separately and a Dubins tree constructed (if it exists). Finally, a shortest of the constructed Dubins trees is selected as a minimum Dubins tree.

In the first step of the algorithm, we nominate any two of the three directed points to play more of a central role, leaving the third directed point to play more of a peripheral role as in the Melzak algorithm for solving Fermat's problem. Having nominated any two of the three directed points to play more of a central role, we draw a distinction between two different types of network topology that can arise: an *even* topology arises when the two circular arcs are similarly oriented, and an *odd* topology arises when the two circular arcs are oppositely oriented. For example, the topology shown in Fig. 6 (right) is even, because both circular arcs, i.e. C components incident to p_1 and p_2, are clockwise oriented. On the other hand, the topology shown in Fig. 6 (left) is odd. Two variants of the algorithm handle the different types of network topology separately. For the most part, both variants of the algorithm operate identically and are illustrated side by side. At the steps where the two variants differ from one another, we will point out the differences. The steps of the algorithm are highlighted below, followed by details of their application and justification of their validity.

Step 1. For each of the three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, choose one of its two incident Dubins circles. Take any two of the three chosen Dubins circles, say, C_1 and C_2 and denote their centres by O_1 and O_2, respectively. Let d denote the distance between the centres O_1 and O_2. In Fig. 6, the third terminal lies in the plane of the page above the line through O_1 and O_2.

Step 2. Construct the auxiliary circle C_{aux} of radius $\frac{1}{2}d\csc\alpha_3$ through the centres O_1 and O_2 and whose centre O_{aux} lies in the plane of the page beneath or above the line through O_1 and O_2 according as α_3 is less than or greater than $\frac{\pi}{2}$. The upper arc of the auxiliary circle, i.e. the segment of the auxiliary circle lying in the plane of the page above the line through O_1 and O_2, is the locus of points at which the lines through O_1 and O_2 meet at an angle of α_3 above the line through O_1 and O_2. To see that this is true, let D denote the midpoint

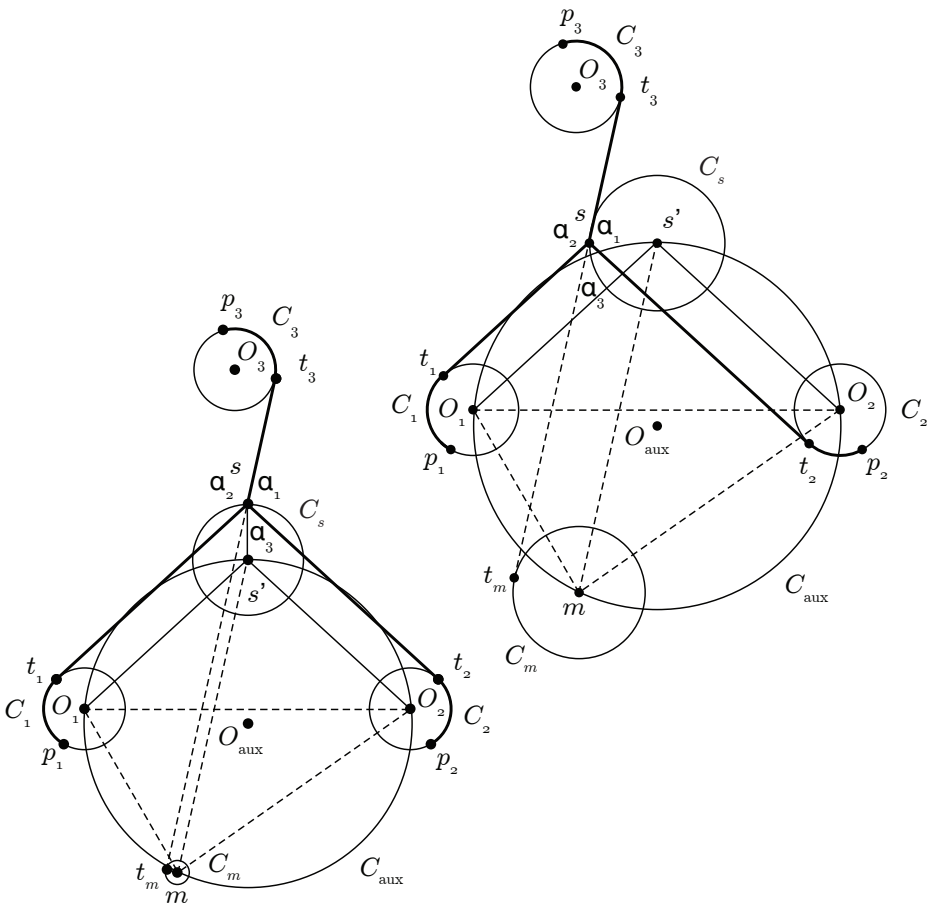


Fig. 6 The Melzak-like construction for the odd topology (left) and even topology (right).

of the straight segment O_1O_2. Then $\triangle O_1DO_{\text{aux}}$ is a (possibly degenerate) right triangle with base O_1D of length $\frac{d}{2}$ and hypotenuse of length $\frac{1}{2}d\csc\alpha_3$. It follows that $\angle O_1O_{\text{aux}}D$ is equal to α_3 or $\pi-\alpha_3$ according as the midpoint D lies in the plane of the page beneath or above the centre O_{aux}. Moreover, the central angle $\angle O_1O_{\text{aux}}O_2$ is twice the angle $\angle O_1O_{\text{aux}}D$. It follows from the Inscribed Angle Theorem that for any point s' of the upper arc of the auxiliary circle C_{aux}, the line segments O_1s' and O_2s' meet at an angle of α_3.

Step 3. Construct the Melzak point m on the lower arc of the auxiliary circle C_{aux} as the third point of the triangle $\triangle O_1O_2m$ with exterior angles α_1, α_2, α_3. Consider a line segment joining the Melzak point m to a variable point s' of the upper arc of the auxiliary circle C_{aux}. By construction, the extension of the line segment ms' beyond s' makes angles α_1 and α_2 with the straight segments $s'O_2$ and $s'O_1$, respectively. To see this, let the position of the point s' vary continuously along the upper arc of the auxiliary circle C_{aux} until it meets the centre O_1 and the line segment ms' coincides with the line segment mO_1. It is easy to see that the extension of the line segment ms' beyond s' makes an angle α_1 with the

straight segment $s'O_2$. By a similar argument, the extension of the line segment ms' beyond s' makes an angle α_2 with the straight segment $s'O_1$.

Step 4. Construct the Melzak circle C_m centred at the Melzak point m and of radius $\rho \csc\left(\frac{\alpha_3}{2}\right) \sin\left(\alpha_1 + \frac{\alpha_3}{2}\right)$ or $-\rho \sec(\frac{\alpha_3}{2}) \cos(\alpha_1 + \frac{\alpha_3}{2})$ according as the network topology is odd or even. We postpone the justification of the choice of radius until we have carried out the remaining steps of the algorithm.

Step 5. Construct the Simpson line that is tangent to the Melzak circle C_m and the third Dubins circle C_3 at the points of tangency t_m and t_3, respectively, with the choice of t_m being decided as follows. In the case of the even topology, the Simpson line $t_m t_3$ should depart from the Melzak circle C_m with the same orientation as the C components of the first and second paths of the network. In the case of the odd topology, the choice of the point of tangency t_m depends on whether the Melzak point m lies in the plane of the page to the left or right of the centre O_{aux}. If the C components of the first and second paths of the network are oriented clockwise and anticlockwise, respectively, then the Simpson line $t_m t_3$ departs from the Melzak circle clockwise or anticlockwise according as the Melzak point m lies to the left or right of the centre O_{aux}. On the other hand, if the C components of the first and second paths of the network are oriented anticlockwise and clockwise, respectively, then the Simpson line $t_m t_3$ departs from the Melzak circle C_m anticlockwise or clockwise according as the Melzak point m lies to the left or right of the centre O_{aux}.

Step 6. Determine the point s' as the other point of intersection of the auxiliary circle C_{aux} and the line through the Melzak point m that is parallel to the Simpson line $t_m t_3$. The point s' is uniquely determined.

Step 7. Construct the Steiner point s as the point of intersection of the Simpson line $t_m t_3$ and the circle C_s centred at the point s' and of radius $\rho \csc\left(\frac{\alpha_3}{2}\right)$ or $\rho \sec\left(\frac{\alpha_3}{2}\right)$ according as the network topology is odd or even, with the choice of s being decided as follows. Generically, there are two points of intersection of the Simpson line $t_m t_3$ and the circle C_s, only one of which can qualify as the Steiner point. Perhaps the easiest way to determine the correct choice is by trying both possibilities and seeing which one produces the desired angles $\alpha_1, \alpha_2, \alpha_3$ between the straight segments of the network, which are to be constructed next. Once again, we postpone the justification of the choice of radius until we have carried out the remaining step of the algorithm.

Step 8. Construct the three paths of type C_s meeting at the Steiner point s. Having constructed the Dubins tree with the chosen orientations of C components (if it exists), we measure its length.

Step 9. Choose a shortest network from the set of eight possible solutions obtained via the previous steps. A shortest of the eight possible networks is a minimum Dubins tree interconnecting the three given directed points in the plane.

Now that we have examined the steps of the algorithm, it remains to justify the choices of the radii of the Melzak circle C_m and the circle C_s. The proof amounts to a demonstration of the fact that the straight segments $t_1 s$, $t_2 s$, $t_3 s$ are parallel to the straight segments $O_1 s'$, $O_2 s'$, ms', respectively, which meet at angles $\alpha_1, \alpha_2, \alpha_3$, according to the third step of the algorithm. We will work backwards from the assumption that these segments are parallel in order to confirm that our choices of radii were indeed correct.

Let us first consider the odd network topology depicted in Fig. 6 (left) with the C components of the first and second paths being oriented clockwise and

anticlockwise, respectively. The angle $\angle s's t_1$ bisects the angle $\angle t_1 s t_2$ and hence is equal to $\alpha_3/2$. Since the straight segments $O_1 s'$ and $t_1 s$ are separated by distance ρ, it follows that the straight segment ss' is of length $\rho \csc\left(\frac{\alpha_3}{2}\right)$, confirming the choice of the radius of the circle C_s. Now consider the trapezoid $ms'st_m$ in Fig. 6 (left). Shorten the straight segments ms' and $t_m s$ at the same rate until the point m coincides with the point s'. The resulting figure is a right triangle with hypotenuse of length $\rho \csc\left(\frac{\alpha_3}{2}\right)$ and base mt_m opposite angle $\pi - \alpha_1 - \alpha_3/2$. This implies that the radius of the Melzak circle C_m is $\rho \csc\left(\frac{\alpha_3}{2}\right) \sin\left(\alpha_1 + \frac{\alpha_3}{2}\right)$. A similar argument applies to the odd network topology when the C components of the first and second paths are oriented anticlockwise and clockwise, respectively.

Let us now consider the even topology depicted in Fig. 6 (right) with the C components of the first and second paths being oriented clockwise. Observe that the angle $\angle O_1 s' s$ is half the difference $\pi - \alpha_3$. Since the straight segments $O_1 s'$ and $t_1 s$ are separated by distance ρ, it follows that the straight segment ss' is of length $\rho \sec\left(\frac{\alpha_3}{2}\right)$, confirming the choice of the radius of the circle C_s. Now denote by l the line through the point s' that is parallel to the line through the points m and t_m. The line l meets the Simpson line $t_m t_3$ at a point p. Observe that the angle $\angle p s' s$ is given by the difference

$$2\pi - \frac{\pi - \alpha_3}{2} - \alpha_3 - \alpha_1 - \frac{\pi}{2} = \pi - \alpha_1 - \frac{\alpha_3}{2}.$$

The right triangle $ps's$ has base ps' equal in length to the radius of the Melzak circle C_m and hypotenuse ss' equal in length to the radius of the circle C_s. It follows that the radius of the Melzak circle C_m is

$$\rho \sec(\frac{\alpha_3}{2}) \cos(\pi - \alpha_1 - \frac{\alpha_3}{2}) = -\rho \sec(\frac{\alpha_3}{2}) \cos(\alpha_1 + \frac{\alpha_3}{2}).$$

A similar argument applies to the even network topology when the C components of the first and second paths are oriented anticlockwise.

The steps of the algorithm are summarised below in Algorithm 1.

Theorem 1 *Consider a full Dubins tree on three terminals* $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$. *Suppose that two terminals, say,* $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *are fixed and the third* $\mathbf{p}_3$ *varied. Then the Steiner point traces out an arc of a limaçon.*

Proof For clarity of exposition, we normalise by setting $\rho = 1$. Given three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ in the plane, suppose that we have determined the angles $\alpha_1, \alpha_2, \alpha_3$ of the network in accordance with the assignment of weights w_1, w_2, w_3. If we fix any two of the three directed points, say, $\mathbf{p}_1$ and $\mathbf{p}_2$ and vary the third, then we trace out a curve in the plane along which the lines through t_1 and t_2 meet at an angle α_3 at a variable Steiner point s. The claim is that this locus of potential Steiner points is an arc of a limaçon, i.e., a polar curve of the form $r = b + a\cos(\theta)$. Before seeking to derive the equation of the limaçon, observe that the equation should be independent of the ratio $\alpha_1 : \alpha_2$, since we are only concerned with the angle α_3 at which the straight segments $t_1 s$ and $t_2 s$ meet. Hence, we are free to choose the special case in which $\alpha_1 = \alpha_2$ and the Melzak point m lies midway between the centres O_1 and O_2 on the lower arc of the auxiliary circle C_{aux} with centre O_{aux}. We will derive the equation of the limaçon corresponding to the odd topology. Later on we will see that the same limaçon includes the loci of potential Steiner points of both types of network topology. Choose a polar coordinate

Algorithm 1: The generalised Melzak algorithm

Input: The three directed points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and the radius of curvature ρ
Output: The optimal location of the Steiner point s and the optimal length of the network

1 For each of the three directed points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, choose one of its two incident Dubins circles.
2 Construct the auxiliary circle C_{aux} of radius $\frac{1}{2}d\csc\alpha_3$ through the centres of the first and second Dubins circles, where d is the distance between the centres.
3 Construct the Melzak point m on the lower arc of the auxiliary circle C_{aux} as the third point of the triangle $\triangle O_1O_2m$ with exterior angles α_1, α_2, α_3.
4 Construct the Melzak circle C_m centred at the Melzak point m and of radius $\rho\csc\left(\frac{\alpha_3}{2}\right)\sin\left(\alpha_1+\frac{\alpha_3}{2}\right)$ or $-\rho\sec(\frac{\alpha_3}{2})\cos(\alpha_1+\frac{\alpha_3}{2})$ according as the network topology is odd or even.
5 Construct the Simpson line that is tangent to the Melzak circle C_m and the third Dubins circle C_3 at the points of tangency t_m and t_3, respectively.
6 Determine the point s' as the other point of intersection of the auxiliary circle C_{aux} and the line through the Melzak point m that is parallel to the Simpson line t_mt_3.
7 Construct the Steiner point s as the point of intersection of the Simpson line t_mt_3 and the circle C_s centred at the point s' and of radius $\rho\csc\left(\frac{\alpha_3}{2}\right)$ or $\rho\sec\left(\frac{\alpha_3}{2}\right)$ according as the network topology is odd or even.
8 Construct the three paths of type CS meeting at the Steiner point s.

system (r,θ) with pole m and polar axis pointing toward O_{aux}. Refer to Fig. 7. Let d denote the distance between the centres O_1 and O_2. Define the point s' and the Steiner point s as in steps 6 and 7 of the generalised Melzak algorithm. Let m' denote the point of the auxiliary circle C_{aux} that is diametrically opposed to the Melzak point m. It follows from the Inscribed Angle Theorem that the angle $m'O_{\text{aux}}s'$ is twice the angle $m'ms$. We know from steps 2 and 7 of the generalised Melzak algorithm that the radii of the auxiliary circle C_{aux} and the circle C_s are $\frac{1}{2}d\csc(\alpha_3)$ and $\csc\left(\frac{\alpha_3}{2}\right)$, respectively. Accordingly, the sum of the three phasors $\overrightarrow{mO_{\text{aux}}}$, $\overrightarrow{O_{\text{aux}}s'}$, and $\overrightarrow{s's}$ is

$$\mathbf{r}(\theta)=\frac{1}{2}d\csc\alpha_3(1,0)+\frac{1}{2}d\csc\alpha_3(\cos 2\theta,\sin 2\theta)+\csc\left(\frac{\alpha_3}{2}\right)(\cos\theta,\sin\theta).$$

After computing the norm of both sides and simplifying, we obtain for the equation of the limaçon

$$r(\theta)=\csc\left(\frac{\alpha_3}{2}\right)+d\csc(\alpha_3)\cos(\theta).$$

Observe that the limaçon is tangent to the Dubins circles at four points of tangency, T_1, T_2, T_3, T_4, which divide it into four differentiable segments. For each combination of orientations of the C components of the first and second paths, we can identify a segment of the limaçon corresponding to the locus of potential Steiner points. For example, if the C components of the first and second paths are oriented clockwise and anticlockwise, respectively, then the first two edges of the network meet at an angle α_3 along the arc of the limaçon between the points of tangency T_1 and T_2. If the C components of the first and second paths are oriented anticlockwise, then the first two edges of the network meet at an angle $\pi-\alpha_3$ along the arc of the limaçon between the points of tangency T_2 and T_3, etc.

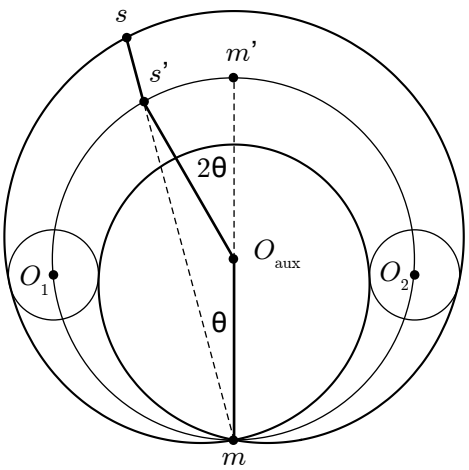


Fig. 7 The graph of the limaçon, i.e., the locus of potential Steiner points.

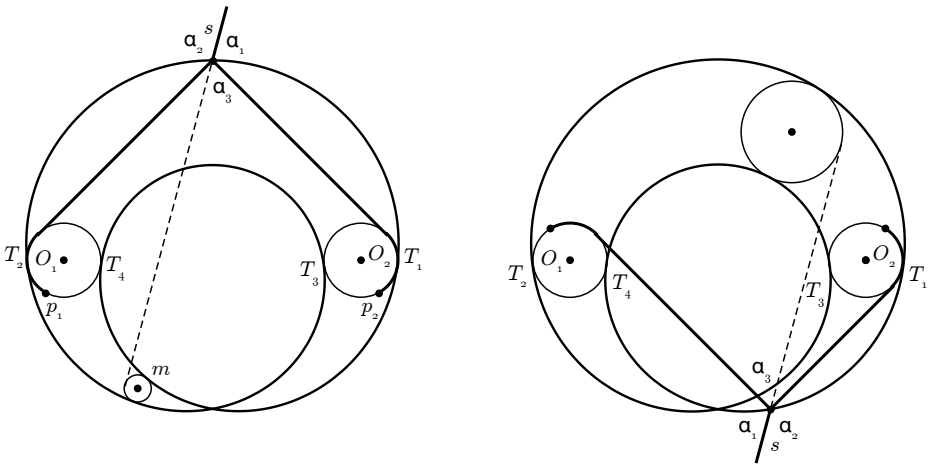


Fig. 8 A full limaçon is associated with both even and odd topologies.

3 Minimal curvature-constrained networks in 3D space

We now present a gradient descent method for numerically approximating the optimal location of the Steiner point in a weighted Dubins tree on three terminals in 3D space. The optimal location of the Steiner point is sought to minimise the total weighted length of the network. The algorithm is not dissimilar from Smith's gradient descent method for constructing Steiner trees in Euclidean d-space [13]. Aside from, of course, the curvature constraint, the main difference is that we choose to minimise a length function, whereas Smith chooses to minimise a potential energy function. Since both objective functions share the same unique critical point, the result is the same. The gradient descent method can be interpreted in terms of a string model in which three strings, representing the three edges of the network, are wrapped around three vertical cylinders and pulled by forces proportional to

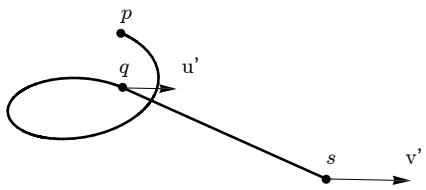


Fig. 9 An edge of the admissible network.

the three weights until the potential energy and, by implication, total weighted length of the network is minimised. Pólya gives a nice description of the string model for Fermat's problem in [11].

Before explaining the steps of the algorithm, we will prove that it converges. As in the planar version of the problem, i.e. Problem 4, we restrict our attention to full Dubins trees. In the 3D version of the problem, i.e. Problem 5, a full Dubins tree on three terminals is a tree, each edge of which is of type CS when traveling from the terminal toward the Steiner point, where C now denotes an arc of a helix of radius ρ. Note that in 3D space, a directed point $\mathbf{p} = (p, \vec{p})$ is associated with two vertical cylinders of radius ρ, both of which are incident to the point p and tangent to the tangent vector $\vec{p}$ based at p. The region of feasible Steiner points therefore consists of $\mathbb{R}^3$ minus the interiors of the six cylinders. Observe that the feasible Steiner point of a full admissible network lies in the interior of the feasible region, which is non-convex. In order to first establish convergence of the algorithm, it is therefore sufficient to demonstrate that the length of an admissible network is locally convex with respect to the position of the Steiner point. To that end, we start with the following preliminary proposition, which is very similar to Proposition 1. The only difference is that the Dubins curve of type CS now lies in 3D space and the arc C is helical. For a curve of type CS in 3D space, we define the transition point as the point where the curve changes from a helical arc to a straight line segment.

Proposition 3 *Suppose that pqs is a curve of type CS with transition point q and that the point p is held fixed while the point s is varied. Then the first variation of length of pqs is the scalar product between the direction of variation of s and the unit vector $\hat{qs}$.*

Proof Let pqs be a curve of type CS where:

- pq is a helical arc of radius ρ.
- q is the transition point.
- qs is a straight segment.

As is illustrated in Fig. 9, the point p is held fixed while the point s moves along the vector of variation $\mathbf{v}'$. The transition point q must move along a vector $\mathbf{u}'$ simultaneously with the point s, for otherwise the constant gradient constraint could be violated at the transition point q. Observe that the vector $\mathbf{u}'$ based at q, specifying the velocity of the point q, is necessarily tangent to the cylinder. Let L denote the length of the curve pqs. Note that the helical arc is just a straight segment wrapped around a cylinder. In other words, we can represent the helical arc by a vector $\vec{pq}$ lying in the plane. Then according to Proposition 1, the first

variation of length of the helical arc pq is given by $\mathbf{u}' \cdot \hat{pq}$, where the perturbation of q in the direction of $\mathbf{u}'$ is tangent to the cylinder. On the other hand, the first variation of length of the straight segment qs is given by $(\mathbf{v}' - \mathbf{u}') \cdot \hat{qs}$. Summing these two terms, we obtain for the first variation of length of pqs

$$L' = \mathbf{u}' \cdot \hat{pq} + (\mathbf{v}' - \mathbf{u}') \cdot \hat{qs} = \mathbf{v}' \cdot \hat{pq}.$$

Theorem 2 *The length of an admissible network is locally convex with respect to the position of the Steiner point.*

Proof By the product rule, the second variation of length of pqs is

$$L'' = \mathbf{v}'' \cdot \hat{qs} + \mathbf{v}' \cdot \hat{qs}'.$$

Note that the first term is zero, because the perturbation of s in the direction of $\mathbf{v}'$ is linear. It follows geometrically that the angle between the vectors $\mathbf{v}'$ and $\hat{qs}'$ is no greater than $\pi/2$. Hence the second variation of L is nonnegative and L is locally convex. Since the weighted sum of locally convex functions is also locally convex, it follows that the length of the network is locally convex with respect to the position of the Steiner point. □

The following gradient descent method numerically approximates the optimal location of the Steiner point in a 3-terminal network in 3D space. Suppose that we are given three directed points in 3D space along with three incident Dubins cylinders, i.e., out-of-bounds regions for the Steiner point (see Fig. 10). We are, of course, assuming that the three terminals are arranged so as to ultimately produce a full Dubins tree, each edge of which is a curve of type CS. Pick any point $s(0)$ lying strictly outside the three Dubins cylinders as an initial feasible Steiner point. It is easy to determine the three helical arcs and three straight segments of the network. Given a fixed point $p = (p_1, p_2, p_3)$ and the feasible Steiner point $s = (s_1, s_2, s_3)$, we need to solve a system of three equations in three unknowns in order to determine two candidate solutions, only one of which may qualify as the point of tangency $q = (q_1, q_2, q_3)$. Note that there will be two solutions lying to either side of the cylinder, the correct choice being made according to a fourth equation. First, suppose that the Dubins cylinder incident to terminal p is centred along axis $\{(c_1, c_2, z) | z \in \mathbb{R}\}$. The magnitude of the normal $n = (q_1 - c1, q_2 - c2, 0)$ to the vertical cylinder at the point of tangency is equal to the minimum turning radius ρ, which implies that

$$n \cdot n = (q_1 - c_1)^2 + (q_2 - c_2)^2 = \rho^2.$$

Second, the dot product of the vector $s - q$ along the straight segment and the normal n to the vertical cylinder at the point of tangency is zero, that is,

$$n \cdot (s - q) = (q_1 - c_1)(s_1 - q_1) + (q_2 - c_2)(s_2 - q_2) = 0.$$

Third, let θ denote the length of the horizontal projection of the helical arc. The gradient of the helical arc must be equal to that of the straight segment, i.e.,

$$\frac{(q_3 - p_3)}{\theta} = \frac{s_3 - q_3}{\sqrt{((s_1 - q_1)^2 + (s_2 - q_2)^2)}}.$$

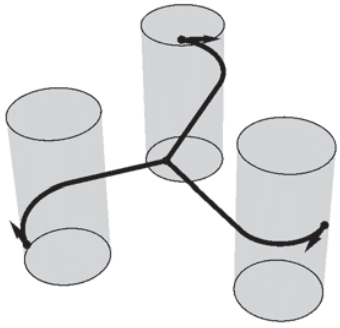


Fig. 10 The minimum curvature-constrained Steiner point algorithm.

These three equations determine two candidate solutions, only one of which is the point of tangency q. In order to eliminate one of the two candidate solutions, we solve a fourth equation, which expresses the fact that the Dubins curve is differentiable at the point of tangency q. Let $R(\theta)$ denote the 2D rotation matrix that rotates vectors counterclockwise by an angle θ. In case the helical arc is left turning from the fixed point p to the point of tangency q, we require that

$$R(\theta)(n_1, n_2) = \frac{(s_1, s_2) - (q_1, q_2)}{\sqrt{((s1 - q1)^2 + (s2 - q2)^2)}}.$$

On the other hand, in case the helical arc is right turning, we replace the angle θ by $-\theta$ in the above equation.

The objective function $L(s)$ gives the total weighted length of the network as a function of the position of the Steiner point s. In order to compute the gradient of the objective function, we simply add the three weighted unit vectors pointing along the straight segments of the network toward the Steiner point. With a slight abuse of notation, we will from now on denote by q_1, q_2, q_3 the three points of tangency of the edges of the network with the three vertical cylinders, respectively. For $i = 1, 2, 3$, let w_i denote the weight assigned to the ith Dubins path of the network and $u_i = s - q_i$ its straight segment. Then the gradient of the objective function is given by

$$\nabla L(s) = w_1\hat{u_1} + w_2\hat{u_2} + w_3\hat{u_3}.$$

To apply a gradient descent method, we take a step downhill in the direction of negative gradient as follows. For some small $\alpha > 0$, set

$$s(i) = s(i-1) - \alpha\nabla L(s(i-1)),$$

and repeat until convergence is obtained.

The steps of the algorithm are summarised below in Algorithm 2.

It is worth pointing out that one could, in principle, obtain a shorter network by omitting from the problem statement the constraint on the horizontal projection of the curvature and in its place imposing only the curvature constraint along with

Algorithm 2: The minimum curvature-constrained Steiner point algorithm

Input: The three directed points p_1, p_2, p_3, the radius of curvature ρ, the step size α and tolerance ϵ

Output: A numerical approximation of the optimal location of the Steiner point s and a numerical approximation of the optimal length of the network

Initialisation: $i = 0$, $s(0) \in \mathbb{R}^3$
repeat
  Determine the three Dubins paths of type CS from the respective terminals toward the Steiner point s.
  Compute the gradient as the sum of the three weighted unit vectors pointing along the straight segments of the network toward the Steiner point.
  $s(i) = s(i-1) - \alpha \nabla L(s(i-1))$
  $i = i + 1$
until $\|s(i) - s(i-1)\| < \epsilon$

the constant gradient constraint on the paths of the network. To see this, note that the curvature k of a helical arc of gradient g on a cylinder of radius r is given by

$$k = \frac{1}{r(1+g^2)}.$$

Observe that in order to satisfy the curvature bound $k \leq \rho^{-1}$, the radius r of a vertical cylinder is allowed to decrease as the gradient of the helical arc to which it is tangent increases. Thus, shrinking the radii of the respective cylinders could result in a further reduction of the length of the network.

In practice, however, there are not one but two constraints on the gradient, namely, a constant gradient constraint and an upper bound of 1/7. Allowing the radius of a helical curve to decrease could result in a violation of the upper bound. To simplify matters in this paper, we have relaxed the upper bound and have used a constraint on the horizontal projection of the curvature in order to obtain a good approximation to a realistic solution.

4 Numerical trials

In this section, we test Algorithm 2 on a set of 10 instances of Problem 5 and compare the numerical results obtained with those produced by a commercially available software tool, namely DOT (Decline Optimisation Tool). The software tool DOT employs a heuristic method. For each instance of Problem 5, the coordinates of the respective terminals are given in meters and the minimum turning radius $\rho = 25$m. The networks constructed satisfy an upper bound of 1:7 on the gradient. All of these values are typical in practice. We set the step-size parameter $\alpha = 1$ and the tolerance $\epsilon = 0.25$m. For the problem instances tested, convergence to within a tolerance ϵ is obtained in less than 30 iterations. See Table 1 for a comparison of results.

Fig. 11 depicts a plot of a network generated by DOT (left) alongside a plot of the corresponding network generated by Algorithm 2 (right). Observe that the network generated by DOT has a T-intersection at which a minor decline meets a main decline at an angle of $\pi/2$. Since the statement of Problem 5 does not include this additional constraint on the junction, it makes sense that the networks generated by Algorithm 2 are shorter than those generated by DOT.

Table 1 Numerical trials

Terminal $\mathbf{p}_1$	Terminal $\mathbf{p}_3$	Terminal $\mathbf{p}_2$	DOT length (m)	Algorithm 2 length (m)
(6, -7, 0, -5π/6)	(-2, 6, 2, -7π/6)	(-7, -7, 1, -5π/6)	724.04	722.75
(5, 0, 0, -7π/3)	(-5, 9, 2, π/6)	(-5, -1, 1, -π/2)	584.62	583.5
(7, -3, 0, -π/2)	(-4, 7, 2, -7π/6)	(-5, -2, 1, -7π/6)	598.04	596.75
(9, -2, 0, -π/2)	(2, 7, 2, -4π/3)	(-4, -7, 1, -4π/3)	681.44	680.25
(4, -8, 0, -π)	(4, 8, 2, -π/3)	(-10, -2, 1, -4π/3)	740.14	739
(8, -6, 0, -π/2)	(4, 7, 2, -2π/3)	(-8, -4, 1, -7π/6)	750.41	749.25
(10, -2, 0, -5π/6)	(4, 6, 2, 0)	(-3, -5, 1, -7π/3)	570.18	558
8, -4, 0, -5π/6)	(-4, 6, 2, -2π/3)	(-6, -1, 1, -7π/6)	556.19	555
(10, 0, 0, -5π/6)	(-4, 8, 2, -π)	(-5, -1, 1, -2π)	600.78	599.5
(10, -2, 0, -π)	(4, 7, 2, 0)	(-5, -5, 1, -11π/6)	612.96	606

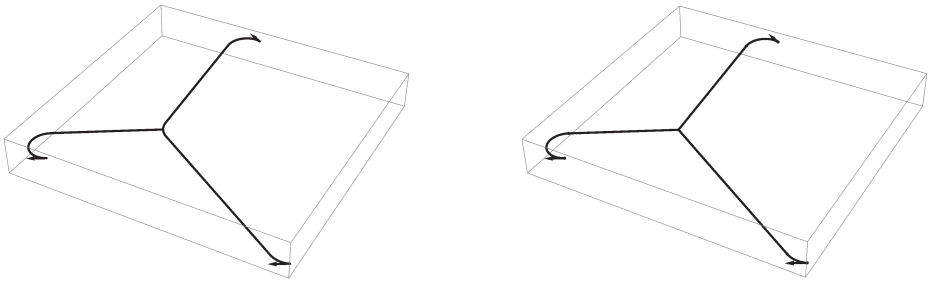


Fig. 11 A network generated by DOT (left) and the corresponding network generated by Algorithm 2 (right).

5 Conclusion

We have given algorithms to find minimal Dubins trees for three directed points in the plane and in 3D space. For applications to mining, it is necessary to also incorporate a gradient constraint so that the latter algorithm is only useful if the resulting network has gradient less than the maximum allowed. In [2], there is a discussion of how to find gradient constrained Steiner trees in 3D space. So a challenge is to merge the methods in this paper and in [2] to find a full solution for the 3D gradient constrained Dubins problem.

For cases of more than 3 given directed points in the plane, the 3-terminal algorithm will generalise as in the Melzak algorithm to construct full Dubins trees, so long as the directed points are suitably arranged in the plane. The edges between the given directed points will be of type CS, whereas those between Steiner points will be straight segments.

References

1. Ayala Hoffmann, J., M. Brazil, J. H. Rubinstein, and D. Thomas, Extendibility and Path Components of Admissible Paths for the Dubins Problem, Australian Control Conference, 440–444 (2011)
2. Brazil, M., J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald, Gradient-constrained minimum networks. I. Fundamentals, Journal of Global Optimization, 139–155 (2001)

3. Brazil, M., Grossman, P. A., Lee, J. H. Rubinstein, D. A. Thomas and N. C. Wormald, Constrained path optimisation for underground mine layout, The 2007 International Conference of Applied and Engineering Mathematics (ICAEM07), London, 856–861, (2007),
4. Dubins, L. E., On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents, American Journal of Mathematics, 79.3, 497–516 (1957)
5. Gilbert, E. N., Minimum Cost Communication Networks, Bell System Technical Journal, 46, 2209–2227 (1967)
6. Hwang, F., D. Richards, and P. Winter, The Steiner Tree Problem, North-Holland, Amsterdam (1992)
7. Melzak, Z. A., On the Problem of Steiner, Canadian Mathematical Bulletin, 4.2, 143–148 (1961)
8. Pólya, G., Induction and analogy in mathematics, Princeton University Press, Princeton (1954)
9. Rubinstein, J. H., and D. A. Thomas, A Variational Approach to the Steiner Network Problem, Annals of Operations Research, 33.6, 481–499 (1991)
10. Smith, W. D., How to find Steiner minimal trees in Euclidean d-space, Algorithmica, 7.2–3, 137–177 (1992)
11. Weng, J. F., Generalized Melzak's Construction in the Steiner Tree Problem, International Journal of Computational Geometry & Applications, 12.6, 481–488 (2002)

Chapter 4

The Hirsch conjecture

This chapter provides some background material on the Hirsch conjecture, which will be further explored in the following two chapters. The Hirsch conjecture concerns polyhedra, familiar examples of which include the tetrahedron, the cube and the other Platonic solids, although polyhedra are not limited to 3D space. Generally speaking, a d-dimensional polyhedron can be defined as the intersection of a finite number of half spaces in $\mathbb{R}^d$. Each polyhedron has a number of faces [24]. The 0-dimensional faces are called vertices, the 1-dimensional faces are called edges and the $(d-1)$-dimensional faces are called facets. Formally, a face of a polyhedron is a subset of points that maximises some linear functional. Now, in this context, the distance between two vertices is measured as the least number of edges required to connect the pair of vertices. The Hirsch conjecture, in its original form, states that any two vertices of a d-dimensional polyhedron with n facets can be joined by a sequence of at most $n-d$ edges [29]. The conjecture is easily verified in the examples mentioned above. However, the conjecture is now known to be false in general due to a counterexample published by Santos in 2010 – a 43-dimensional polytope with 86 facets [43]. That being said, the problem of obtaining a larger linear or polynomial upper bound for the diameter of a polyhedron remains open. In this chapter, we will discuss the motivation for studying the Hirsch conjecture before summarising several key results that have been obtained in relation to the conjecture.

4.1 Linear programming

The Hirsch conjecture is primarily of interest due to its implications for the computational efficiency of the simplex method, an algorithm invented by George Dantzig at the RAND Corporation in 1947 for the solution of linear programming problems [16]. The simplex method is widely regarded as one of the most important algorithms of all time, finding many applications

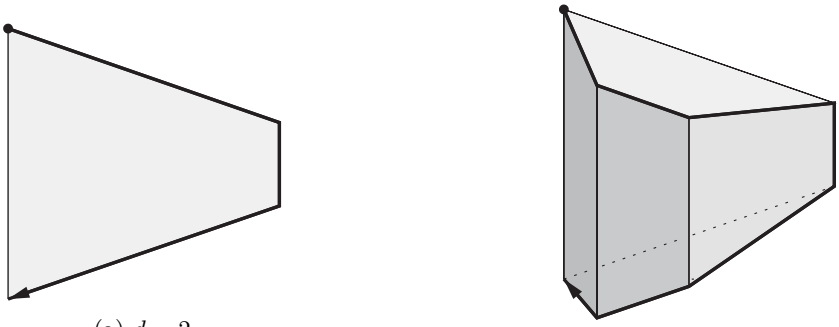
in industry, where it is important to, for example, maximise profit or minimise cost subject to various budgetary or resource constraints [14]. We saw an example of an application of linear programming in the form of a scheduling problem in chapter 1. Other examples of applications of linear programming include portfolio selection, blending problems, transhipment problems, assignment problems, shortest-route problems, etc [2]. Stated mathematically, the problem of linear programming is to minimise a linear cost function subject to a set of linear equality and inequality constraints. That is, given as input an $(n-d) \times n$ real matrix $\mathbf{A}$, a resource vector $\mathbf{b} \in \mathbb{R}^{n-d}$ and a cost vector $\mathbf{c} \in \mathbb{R}^n$,

$$\begin{aligned} \text{minimise } & \mathbf{c} \cdot \mathbf{x} \\ \text{subject to } & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$. Let $\mathbf{a}_i$ denote the vector comprising the entries of the ith row of the matrix $\mathbf{A}$. Each of the n constraints of the form $x_i \geq 0$ carves out a closed half space from d-dimensional Euclidean space. Hence, the feasible region P is a convex polyhedron of dimension d with n facets. In the case where P is bounded, it is referred to as a *polytope*. If the polyhedron has at least one vertex, then it is called a *pointed* polyhedron. If each vertex of a pointed d-dimensional polyhedron is of degree d, then the polyhedron is said to be *simple*. On the other hand, if each facet of a d-dimensional polytope has exactly d vertices, then the polytope is *simplicial*.

4.2 A physical model for the solution of a linear programming problem

Picture a 3D box, the sides of which correspond to the facets of a polyhedron. Suppose that you were to place a marble inside the box and to allow it to roll downhill along the surface of the box. Under the force of gravity, the marble would eventually arrive at a minimum point if one existed. In this case, the linear cost function to be minimised is height or potential energy. The example illustrates the mechanics of the simplex method with one restriction. The simplex method is restricted to descend along the edge graph or 1-skeleton of the polyhedron and cannot cross the interior of any higher dimensional face [12]. The simplex method is a combinatorial version of a gradient descent method, which is initialised at some vertex of the polyhedron and proceeds by pivoting along an edge toward any adjacent vertex that allows for an improvement to the value of the objective function to be made. Such a descending path is referred to as a *monotone path*. At each vertex of the path, there is a choice amongst at least $d-1$ edges along which to proceed. The choice is made according to some pivot rule, e.g., steepest edge, random edge, largest increase, largest coefficient, Bland's rule, etc. Since each pivot requires a number of arithmetic operations that is polynomial in d and n, the computational efficiency of the simplex



(a) $d = 2$ (b) $d = 3$

Figure 4.1: The Klee Minty cubes in dimensions 2 and 3.

method depends on the length of the path, i.e., the number of pivot steps, which is affected by the selection of pivot rule.

4.3 Pathological examples

It came as somewhat of a surprise when in 1969, Klee and Minty published a family of linear programming problems with d variables and $2d$ constraints for which the simplex method when equipped with Dantzig's original pivot rule requires a number of pivots that grows exponentially in d [33]. The set of feasible solutions is a deformed d-dimensional hypercube, called a *Klee-Minty cube*. The cost vector is oriented so that the simplex method with Dantzig's largest coefficient pivot rule passes through all 2^d vertices and hence requires $2^d - 1$ pivot steps. See Fig. 4.1 for illustrations of the Klee-Minty cube for dimensions 2 and 3. The algorithm traverses the path marked with a thick line. This family of linear programming problems indicates that the simplex method has worst-case exponential complexity when equipped with Dantzig's largest coefficient rule. Now, observe that in Fig. 4.1 the first and last vertex in the edge-path are adjacent to one another. The simplex method could terminate with a single pivot under a different pivot rule. This observation leads to the question of whether for any possible pivot rule, there exists a family of linear programming problems for which the simplex method requires an exponentially growing number of pivot steps. For several pivoting rules, similar families of deformed products have been constructed [1]. The question is closely connected to the problem of whether linear programming can be performed in strongly polynomial time, listed amongst Smale's "mathematical problem for the next century" [44].

4.4 Diameters of polyhedra

The preceding discussion of edge-paths in a polyhedron motivates the following definitions of length. Given two vertices $\mathbf{v}$ and $\mathbf{w}$ of a polyhedron P, we define the *distance* $\delta_P(\mathbf{v}, \mathbf{w})$ as the least number of edges required to join $\mathbf{v}$ and $\mathbf{w}$. This number always exists, as the vertices and bounded edges of P form a connected graph, assuming that the supporting hyperplanes are in general position. Following Klee, for each vertex $\mathbf{v}$ of P, we define the $\mathbf{v}$-*radius* of P as

$$\rho_{\mathbf{u}}(P) = \max\{\delta_P(\mathbf{v}, \mathbf{w}) | \mathbf{w} \in \text{vert}(P)\}.$$

The *radius* of P is defined as

$$\rho(P) = \min\{\rho_{\mathbf{v}}(P) | \mathbf{v} \in \text{vert}(P)\},$$

whereas the *diameter* of P is defined as

$$\delta(P) = \max\{\rho_{\mathbf{v}}(P) | \mathbf{v} \in \text{vert}(P)\}.$$

We note that

$$\rho(P) \leq \delta(P) \leq 2\rho(P).$$

Finally, we define $\Delta(d, n)$ as the maximum of $\delta(P)$ as P ranges over all (possibly unbounded) d-dimensional polyhedra with n facets. Similarly, we define $\Delta_b(d, n)$ as the maximum of $\delta(P)$ as P ranges over all bounded d-dimensional polyhedra, i.e., polytopes, with n facets. The quantity $\Delta(d, n)$ gives a performance estimate for the simplex method when equipped with the best possible pivot rule, call it the "clairvoyant's rule", which always selects the shortest monotone path toward an optimum vertex.

4.5 A brief history of the Hirsch conjecture

In 1957, Warren M. Hirsch communicated to George Dantzig the following conjecture [46]:
Conjecture 4.1 (Hirsch conjecture)**.**

$$\Delta(d, n) \leq n - d.$$

The intuition behind the conjecture is as follows. Given a simple d-dimensional polyhedron P with n facets, suppose that one traverses a path joining diametrically opposed vertices. The initial vertex is incident to exactly d facets. The traversal of an edge involves discarding an old facet and exchanging it for a new facet. If the path never revisits a facet, then the diameter can be no greater than $n - d$.

In 1993, Kalai and Kleitman published a very short paper containing a 'pseudopolynomial" upper bound, which is the best known upper bound to date [27]:

$$\Delta(d, n) < n^{\log_2(d)+2}.$$

The upper bound grows slower than exponentially, but faster than any polynomial. Then, in 2010, Santos disproved the bounded Hirsch conjecture, publishing a 43-dimensional counterexample with 86 facets. Later, in 2012, Matschke, Weibel and Santos improved Santos' original result, obtaining a lower dimensional counterexample of dimension 20 [39]. Today, the problem of whether there exists a polynomial or even linear upper bound for $\Delta(d, n)$ remains open.

Before closing this chapter, we examine one other result which will be used in the next chapter. Via an inductive argument, Klee reduces the Hirsch conjecture to an equivalent conjecture called the d-step conjecture, which states that any two vertices of a polytope whose number of facets is equal to twice its dimension d can be connected by an edge-path of length no greater than d [32]. Consider, for example, the cube and "wedge" over a pentagon, which are illustrated in Fig. 4.2.

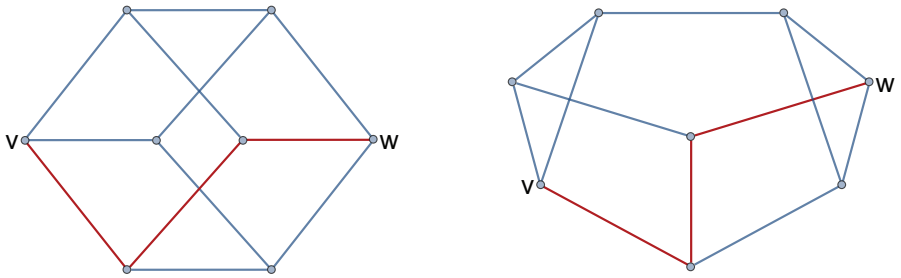


Figure 4.2: The cube and wedge over a pentagon.

Theorem 4.2 (d-step conjecture)**.**

$$\Delta_b(d, n) \leq n - d \quad \textit{for all} \quad d, n \qquad \textit{if and only if} \qquad \Delta_b(d, 2d) \leq d \quad \textit{for all} \quad d.$$

There are two sides to the inductive argument. On the one hand, suppose that the number of facets of a polytope P is less than twice its dimension. Since each member of a pair of diametrically opposed vertices is incident to at least d facets, it follows that the two diametrically opposed vertices in question lie in a common facet F. Observe that the facet F is of dimension $d-1$ and itself has no more than $n-1$ facets. If the diameter of the facet F is bounded from above by $(n-1)-(d-1)$, then

$$\operatorname{diam}(P) \leq \operatorname{diam}(F) \leq (n-1)-(d-1) = n-d.$$

According to this "downward" inductive argument, it suffices to restrict one's attention to poly-

topes for which $n \geq 2d$. By the same line of reasoning, it is clear that we may focus our attention on polytopes for which no two diametrically opposed vertices lie in a common facet. On the other hand, suppose that the number of facets of a polytope P is greater than twice the dimension. Then we can build a new polytope Q of one dimension more and with one more facet via the wedge construction. To form a wedge, start by constructing the prism $P \times [0, 1]$. Then collapse one of the lateral facets $F \times [0, 1]$. The resulting polytope is a wedge Q over P with foot F. See Fig. 4.3 for an illustration. Observe that any path in the wedge Q projects onto a path in the original polytope P. Accordingly, if the diameter of the wedge Q is bounded from above by $(n + 1) - (d + 1)$, then

$$\operatorname{diam}(P) \leq \operatorname{diam}(Q) \leq (n + 1) - (d + 1) = n + d.$$

The "upward" and "downward" inductive arguments together imply that it is sufficient to restrict one's attention to polytopes for which the number of facets is equal to twice the dimension. Polytopes for which the number of facets is equal to twice the dimension and for which no two diametrically opposed vertices lie in a common facet are called *Dantzig figures*.

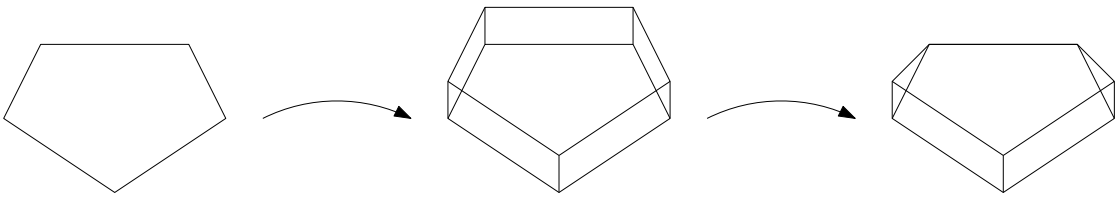


Figure 4.3: The wedge construction.

Chapter 5

The Pachner graph of dual Dantzig figures

5.1 Introduction

In this chapter, we present an algorithm for enumerating spherical triangulations, a class of objects that encompasses the duals of Dantzig figures, i.e., polytopes that are isomorphic to Dantzig figures. Such an algorithm could be used to verify the d-step conjecture in some low dimensional cases. In particular, we produce catalogues of spherical triangulations that include all dual Dantzig figures in dimensions 3, 4 and 5, respectively, and verify the d-step conjecture in these cases. We note that the d-step conjecture has been verified for $d \leq 6$ [9]. The approach is to generate the so-called Pachner graph, where each node represents a dual Dantzig figure and each edge represents a move called a bistellar flip for passing between dual Dantzig figures [6]. The approach that we adopt is similar, albeit significantly faster, than the approach adopted in a paper by Koranne and Kulkarni [35]. The differences are that in our case, we work with simplicial as opposed to simple polytopes, we represent a polytope by its incidence matrix as opposed to its entire face poset and we perform dynamic pruning to a greater extent as we search for new polytopes.

5.2 Dual polytopes

It will be convenient to work in the category of dual simplicial polytopes, because then we can use a well understood set of moves called bistellar flips for generating "combinatorially adjacent" polytopes from existing polytopes [17]. To construct the dual polytope P^* of a full-dimensional

polytope P in $\mathbb{R}^d$, start by performing a translation if necessary, so that the polytope P contains the origin in its interior. The dual polytope P^* is defined by

$$P^* = \{\mathbf{y} | \mathbf{x} \cdot \mathbf{y} \leq 1 \text{ for all } \mathbf{x} \in P\},$$

where $\mathbf{x} \cdot \mathbf{y}$ denotes the Euclidean inner product of the vectors $\mathbf{x}$ and $\mathbf{y}$. Note that the operation of constructing the dual of a polytope reverses the order of inclusion of its face poset. In particular, if P is a simple polytope, then P^* is a simplicial polytope. Moreover, an edge-path in P comprising steps between vertices corresponds to a ridge-path in P^* comprising steps between facets.

5.3 Bistellar flips

Generally speaking, a bistellar flip on a triangulated $(d-1)$-manifold M involves the exchange of a set of k mutually adjacent maximal simplices for another set of $d+1-k$ mutually adjacent maximal simplices, where the two sets together form the boundary of a d-simplex [36].

Definition. *Let M be a triangulated $(d-1)$-manifold. Suppose that M is not a simplex and that M includes several mutually adjacent maximal simplices, which constitute part of the boundary of a d-simplex denoted by B. A bistellar flip ϕ, i.e., Pachner move, on M is defined by*

$$\phi(M) = (M \setminus \partial B) \cup (\partial B \setminus M).$$

Suppose that we perturb the vertices of a simplicial polytope P. The combinatorics of P change whenever $d+1$ vertices pass through a supporting hyperplane. To see this, consider a d-simplex on $d+1$ vertices, several facets of which belong to the boundary of the polytope P, while the remaining facets lie inside P. If the $d+1$ vertices pass through a supporting hyperplane, then the d-simplex collapses before reversing orientation. After reversing orientation, those facets of the d-simplex which initially lay inside the polytope P now emerge as facets belonging to its boundary and vice versa. It is easy to see that we can transform any two d-dimensional simplicial polytopes P and P', each of which comprises n vertices, into one another via a sequence of such bistellar flips. Denote the vertices of P by $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ and those of P' by $\mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_n$. We may assume that n is greater than $d+1$, for otherwise both P and P' are simplices. For each pair of vertices $\mathbf{v}_i$ and $\mathbf{v}'_i$, we can construct a linear homotopy $(1-t)\mathbf{v}_i + t\mathbf{v}'_i$, where $t \in [0,1]$. Suppose that we homotope each of the n vertices of P one by one and consider, in particular, the ith vertex $\mathbf{v}_i$. If the two polytopes P and P' are in general position with respect to one another, then the linear homotopy of the vertex $\mathbf{v}_i$ will not pass through an affine subspace defined by fewer than d of the $n-1$ other vertices of P. That is, the linear homotopy of the vertex $\mathbf{v}_i$ will only pass through affine subspaces that are defined by exactly d of the $n-1$ other vertices of P. If a transition of the combinatorics of P occurs due to such a homotopy, then it corresponds to a bistellar flip on the boundary of P.

5.4 A Matveev-like result for dual Dantzig figures

We can make an even stronger claim regarding dual Dantzig figures, namely, that any two d-dimensional dual Dantzig figures can be transformed into one another via a sequence of bistellar flips that excludes $(1, d)$- and $(d, 1)$-flips. This result is useful, because it allows us to narrow the search space and hence obtain a faster algorithm for generating the Pachner graph of dual Dantzig figures. The result is analogous to Matveev's theorem, which states that any two triangulations of a 3-manifold with at least two tetrahedra can be connected via a sequence of (2, 3)- and (3, 2)-flips [40]. We start with a preliminary proposition.

Proposition 5.1. *Suppose that a polytope P has two nonadjacent facets F and F'. Then there exists a projective transformation π placing F and F' in parallel hyperplanes.*

Proof. The polytope P is embedded in the space E. See Figure 5.1. The facets F and F' lie in supporting hyperplanes H and H', respectively. The hyperplanes H and H' intersect in an affine subspace J. Choose a hyperplane K that separates P from J. Construct $E \times \mathbb{R}$, the product of E with the real numbers. Choose a point u in $J \times (\mathbb{R} \sim 0)$ as the point of projection. Project onto the hyperplane $K \times \mathbb{R}$ lying in $E \times \mathbb{R}$. The facets $\pi(F)$ and $\pi(F')$ lie in parallel hyperplanes in $K \times R$. □

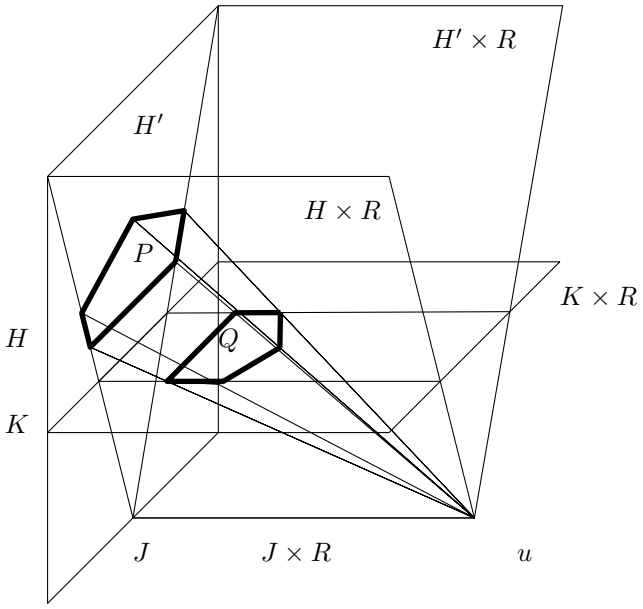


Figure 5.1: A projective transformation places the facets F and F' in parallel hyperplanes.

Theorem *Any two d-dimensional dual Dantzig figures can be transformed into one another via a sequence of bistellar flips that excludes $(1, d)$- and $(d, 1)$-flips.*

Proof. Consider a pair P and P' of d-dimensional dual Dantzig figures. If necessary, scale the two polytopes and perform projective transformations according to Proposition 1, so that the top facets F and F' lie in a hyperplane H, while the bottom facets G and G' lie in a parallel hyperplane I. Apply an orientation preserving affine transformation to the hyperplane H that transforms F into F'. Likewise apply an orientation preserving affine transformation to the hyperplane I that transforms G into G'. Clearly, the vertices of the polytope P continue to be extreme points during the transformations described above, as is the case for the polytope P'. On the other hand, no new vertices are introduced. It follows that no $(1, d)$- or $(d, 1)$-flips take place. □

5.5 The Pachner graph algorithm

We now present an algorithm for generating all dual Dantzig figures in a given dimension. In particular, the algorithm generates a connected component of the Pachner graph that is rooted at the cross polytope and includes all dual Dantzig figures in addition to a number of triangulated spheres that cannot be realised as polytopes. To be clear, Pachner moves replace a given triangulation of a piecewise linear manifold, e.g., a triangulated sphere, by a different triangulation of a homeomorphic manifold, but may not preserve realizability as polytopes. For example, it is known that there are triangulated 3-spheres with 8 vertices which cannot be realised as polytopes [23]. As we will see, we can obtain such triangulated spheres from the 4-dimensional cross-polytope via sequences of Pachner moves. Moreover, Walkup found a triangulation of a 27-sphere which violates the Hirsch bound [45].

Step 1. Construct an incidence matrix $\mathbf{I}(C)$ *for the* d*-dimensional cross-polytope* C*.* Each triangulation $\mathcal{T}$ is represented by an incidence matrix $\mathbf{I}(\mathcal{T})$, which includes a row for each vertex v and a column for each facet F of $\mathcal{T}$. In particular, $(v, F) = 1$ if v is incident upon F and $(v, F) = 0$ otherwise. For example, an incidence matrix, which is not unique, for the octahedron is

$$\mathbf{I}(C) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Step 2. Enumerate the collection $\mathcal{F}(C)$ *of all flip-out sets associated with the cross-polytope* C*.* Recall that performing a bistellar flip on a triangulation $\mathcal{T}$ involves the exchange of one half of the boundary ∂B of a d-simplex, that is, $\partial B \cap \mathcal{T}$, with the other half of ∂B, that is, $\partial B \setminus \mathcal{T}$. We

refer to the maximal simplices in $\partial B \cap \mathcal{T}$ as the *flip-out set* f and those in $\partial B \setminus \mathcal{T}$ as the *flip-in set* f'. The flip-out set comprises a number of mutually adjacent $(d-1)$-simplices. Regarding the cross-polytope, it is easy to see that the mutually adjacent simplices come in pairs. Two $(d-1)$-simplices are adjacent if they share $d-1$ vertices, which can easily be checked by computing the inner product of the two corresponding columns of $\mathbf{I}(C)$. For example, the cardinality of the collection of flip-out sets $\mathcal{F}(C)$ associated with the octahedron is 12, that is, to each edge of the cube there corresponds a flip-out set.

Step 3. Generate the isomorphism signature $\sigma(C)$ of the cross-polytope C. The isomorphism signature determines the isomorphism class for a simplicial polytope $\mathcal{T}$, that is, the class of all simplicial polytopes that can be obtained by relabelling the vertices of $\mathcal{T}$. In the interests of designing an algorithm that is not very slow, it is clearly desirable to discard from consideration any simplicial polytope $\mathcal{T}'$ that is equivalent up to a relabelling of its vertices to a previously generated simplicial polytope $\mathcal{T}$. A number of ingredients are required in order to guarantee that the signature correctly identifies the isomorphism class of a simplicial polytope. First, we invoke a result of Kalai, which says that the face lattice of a simple polytope can be reconstructed from its graph [26]. It follows that the face lattice of a simplicial polytope can be reconstructed from its dual graph. What this means is that it is sufficient to compare the dual graphs of two simplicial polytopes in order to determine whether or not they are isomorphic. In other words, we can avoid having to compare incidence matrices in order to test for isomorphism. Note that we do not make the same claim for triangulations that cannot be realised as polytopes. Now, it is easy to generate the adjacency matrix $\mathbf{A}(\mathcal{T})$ corresponding to the dual graph of a triangulation $\mathcal{T}$. If two simplices of $\mathcal{T}$ share $d-1$ vertices, i.e., a ridge, then the corresponding vertices of the dual graph are connected by an edge. With this in mind, consider the symmetric matrix $\mathbf{B}(\mathcal{T}) = \mathbf{I}(\mathcal{T})^T\mathbf{I}(\mathcal{T})$. Define

$$a_{ij} = \begin{cases} 1 & \text{if } b_{ij} = d-1 \\ 0 & \text{otherwise.} \end{cases}$$

From the adjacency matrix $\mathbf{A}(\mathcal{T})$ of $\mathcal{T}$, we can construct its dual graph $g(\mathcal{T})$ and then, using a test for graph isomorphism, infer whether or not two simplicial polytopes are isomorphic. To see this, pick out a canonical member $g_c(\mathcal{T})$ from the isomorphism class of $g(\mathcal{T})$ [11]. Let $\mathbf{A}_c(\mathcal{T})$ denote the adjacency matrix of the canonical graph $g_c(\mathcal{T})$, which can be different from $\mathbf{A}(\mathcal{T})$. Moreover, let $\mathbf{v}_c(\mathcal{T})$ denote the bit string that is obtained by flattening out the entries of $\mathbf{A}_c(\mathcal{T})$, in other words, concatenating consecutive rows of $\mathbf{A}_c(\mathcal{T})$. Then the signature of $\mathcal{T}$ is given by

$$\sigma(\mathcal{T}) = \mathbf{v}_c(\mathcal{T}).$$

The signature provides us with an efficient means for sorting and comparing triangulations. A test for polytope isomorphism reduces to a comparison of two signatures.

To reiterate, the result of Kalai has been established only for simplicial polytopes as opposed to triangulated spheres in general. We cannot be sure that the use of the signature will not result

in us conflating two or more isomorphism classes of triangulated spheres. However, we are here concerned with measuring the diameters of dual Dantzig figures and do not care whether or not we fail to account for all triangulated spheres that are unique up to isomorphism. To summarise, the information that we retain about any triangulation $\mathcal{T}$ that we obtain includes its incidence matrix $\mathbf{I}(\mathcal{T})$, collection $\mathcal{F}(\mathcal{T})$ of flip-out sets, and signature $\sigma(\mathcal{T})$.

Step 4. For each triangulation $\mathcal{T}$ in level i and for each flip-out set f in $\mathcal{F}(\mathcal{T})$, attempt to construct a new triangulation $\mathcal{T}'$ in level $i+1$ by performing a bistellar flip. Given a flip-out set f in $\mathcal{F}(\mathcal{T})$ with vertex set $\mathrm{vert}(f)$, it is easy to represent the maximal simplices of the d-simplex B on $\mathrm{vert}(f)$ by column vectors as in the incidence matrix $\mathbf{I}(\mathcal{T})$. We can in turn list the column vectors corresponding to the flip-in set f', which comprises those maximal simplices in the complement $\partial B \setminus f$. Observe that the k' maximal simplices of f' intersect to form a $d-k'$ simplex, which we denote by F. In order for this particular bistellar flip to be a legal move, it must satisfy the following sufficient condition. The $k'+1$ vertices of F cannot all be contained in a facet of $\mathcal{T}$. This condition is easy to check by computing the inner product of a column vector representing the simplex F with each of the column vectors representing the respective facets of $\mathcal{T}$. If the move is legal, then we construct the incidence matrix $\mathbf{I}(\mathcal{T}')$ from $\mathbf{I}(\mathcal{T})$. That is, we delete the column vectors corresponding to the maximal simplices of f and introduce column vectors corresponding to the maximal simplices of f'.

Step 5. Check if the symmetric matrix $\mathbf{B}(\mathcal{T}')$ could possibly correspond to a dual Dantzig figure. Recall that a dual Dantzig figure contains two facets that do not intersect one another. Now, the symmetric matrix $\mathbf{B}(\mathcal{T}) = \mathbf{I}(\mathcal{T})^T\mathbf{I}(\mathcal{T})$ records the respective numbers of vertices shared by pairs of facets of a simplicial polytope $\mathcal{T}$ or simplicial sphere. In particular, $\mathbf{B}(\mathcal{T})$ contains a zero entry if $\mathcal{T}$ contains two nonintersecting facets. Thus, we may choose to inspect $\mathbf{B}(\mathcal{T})$ for zero entries in order to filter out some triangulations from the Pachner graph.

Step 6. Generate the signature $\sigma(\mathcal{T}')$ and use binary search to determine whether or not the dual graph of $\mathcal{T}'$ is isomorphic to any of the previously generated dual graphs. Suppose that we have a sorted list of m previously generated signatures. Then we can test if any newly generated signature is a duplicate in $O(\log(m))$ time. If we generate a new signature, then we proceed to the next step.

Step 7. Update the collection of flip-out sets to include or exclude, respectively, any flip-out set that has been created or destroyed by the introduction of the flip-in set f'. We delete from the collection $\mathcal{F}(\mathcal{T})$ of flip-out sets the flip-out set f along with any other flip-out set f'' such that $f \cap f'' \neq \emptyset$. To be precise, any flip-out set f'' is destroyed if it shares a maximal simplex with the flip-out set f. (On the other hand, a flip-out set f'' may not be destroyed if it shares a lower dimensional simplex such as a ridge with the flip-out set f.) Denote by $N(f)$ the set of facets of $\mathcal{T}$ that are adjacent to those in f. Any subset of mutually adjacent facets in $N(f) \cup f'$ qualifies as a new flip-out set, which is added to the collection of flip-out sets.

Step 7. Repeat steps 4-7 until no new triangulation can be generated. The algorithm terminates when no new triangulation with a unique signature can be generated.

Example 5.2. Fig. 5.2 shows the successive layers of the Pachner graph for dimension $d = 3$ arranged in ascending order from top to bottom, where each graph corresponds to the dual of a triangulated sphere on six vertices. There are two graphs, namely that of the cube and that of the wedge over a pentagon.

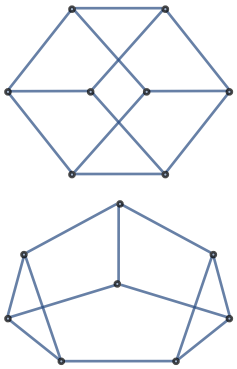


Figure 5.2: The Pachner graph for dimension $d = 3$.

Example 5.3. Fig. 5.3 shows the successive layers of the Pachner graph for dimension $d = 4$ arranged in ascending order from top to bottom, where each graph corresponds to the dual of a triangulated sphere on eight vertices. We reproduce a result of Brückner, establishing the existence of 39 triangulations of the 3-sphere on 8 vertices [10]. Due to the work of Grünbaum and Sreedharan, it is known that only 37 of these triangulated spheres can be realised as polytopes [23]. The 33 graphs illustrated in blue are those of Dantzig figures.

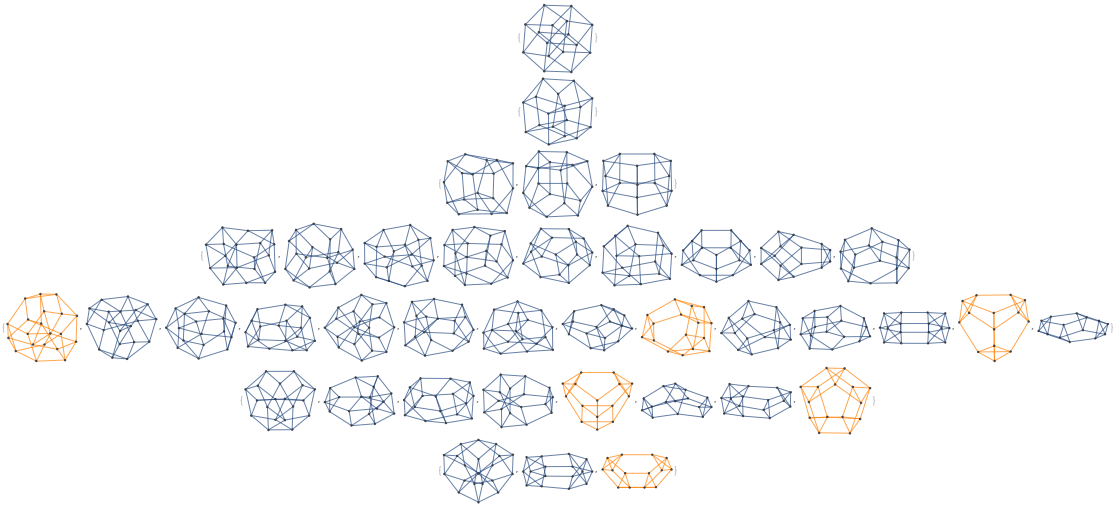


Figure 5.3: The Pachner graph for dimension $d = 4$ with the graphs of Dantzig figures in blue.

Example 5.4. Fig. 5.4 shows the number of combinatorially "Dantzig" triangulations with unique signatures in each of the 17 levels of the connected component of the Pachner graph that is rooted at the 5-dimensional cross polytope. There are 489,446 triangulations in total, all with diameter 5.

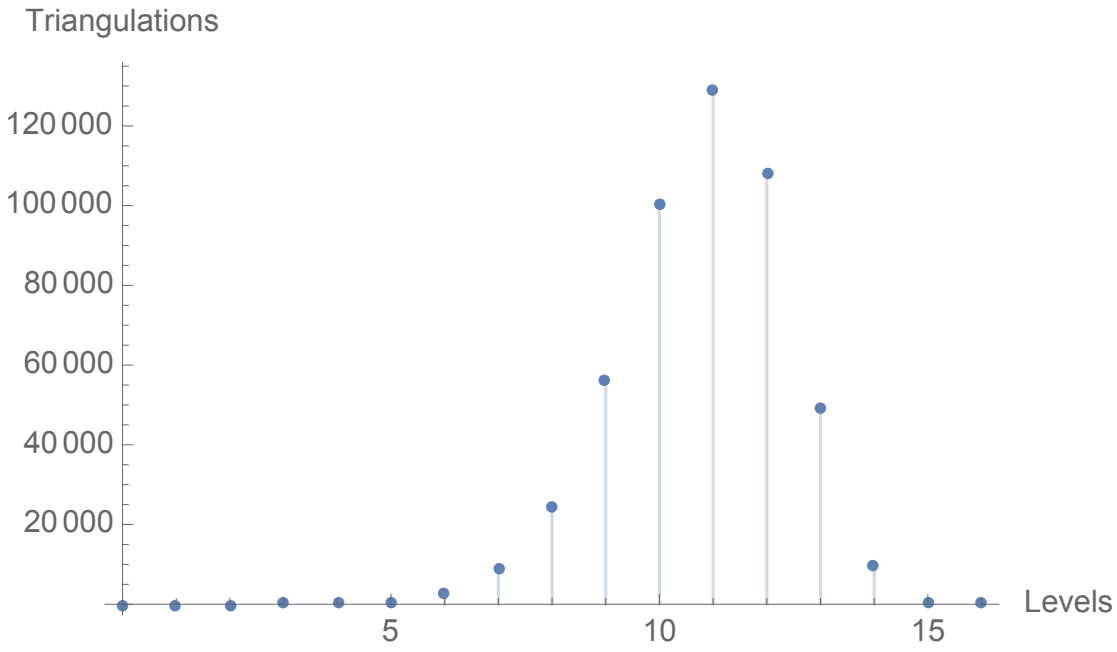


Figure 5.4: The number of combinatorially "Dantzig" triangulations with unique signatures in a connected component of the Pachner graph for dimension $d = 5$.

5.6 Bounding the radius of the Pachner graph

It is easy to see that each bistellar flip on the boundary of a simplicial polytope P changes the length of a ridge path and hence the dual diameter by at most one. To see this, consider two facets F_1 and F_2 of P, each of which is adjacent to some flip-out set f. If F_1 and F_2 are separated by one maximal simplex in the flip-out set f, then they are separated by two maximal simplices in the flip-in set f' and vice versa. In light of this fact, one might attempt to obtain a bound for the dual diameter of a dual Dantzig figure by bounding the radius of the connected component of the Pachner graph to which is belongs, i.e., the number of levels.

However, some examples suggest that the number of levels of the Pachner graph is superpolynomial in d. Consider the last graph of the Pachner graph for dimension $d = 3$ alongside the last graph of the Pachner graph for dimension $d = 4$ (see Fig. 5.5). Both graphs correspond to wedges, which can be constructed as follows. Let Δ_d denote the d-dimensional simplex and P_n

denote the polygon with n facets. Consider the product

$$Q_d = \Delta_{d-2} \times P_{d+2}.$$

The d-dimensional polytope Q_d contains $d+2$ prisms as facets. If we collapse one of the prisms, then we obtain a d-dimensional wedge W_d with $d(d-1)+2$ vertices, i.e., a number of vertices that grows quadratically in d. On the other hand, the d-dimensional hypercube contains 2^d vertices, a number of vertices that grows exponentially in d. Since each bistellar flip can change the number of facets of triangulation by at most $d-1$, i.e., a number of facets that is linear in d, it follows that the Pachner graph cannot have radius bounded from above by some polynomial in d. However, it is worth noting that the wedge W_d is a Dantzig figure only in the case where $d = 3$. If $d > 3$, it is not the case that the wedge W_d contains two vertices having no facets in common. So we cannot immediately draw any conclusions about the radius of the Pachner graph of combinatorially "Dantzig" triangulations from these examples (refer to Step 5 of the algorithm).

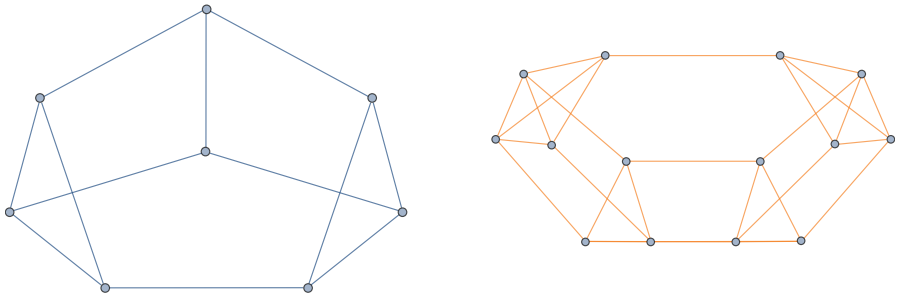


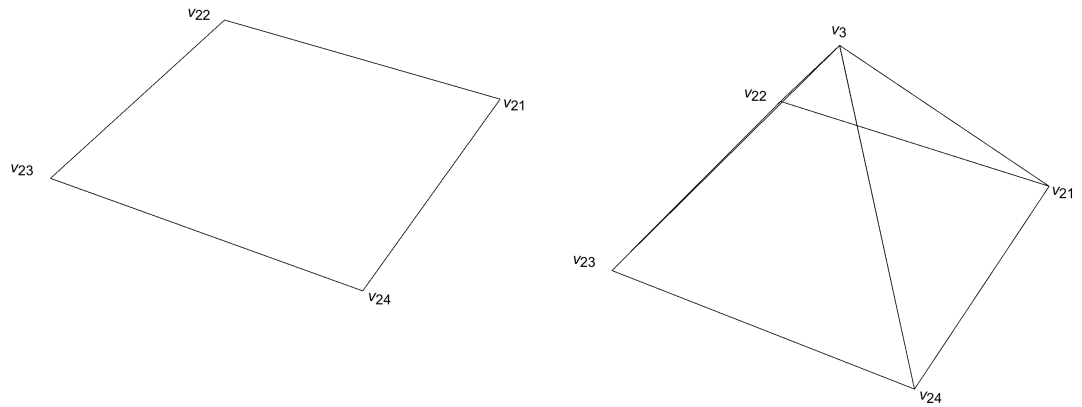
Figure 5.5: The wedges W_3 and W_4 contain numbers of vertices that grow quadratically in d.

Chapter 6

The Hirsch conjecture for lifted polyhedra

The idea of this chapter is to take an arbitrary linear programming problem and to construct an equivalent linear programming problem for which the associated polyhedron has small diameter. Since the two linear programming problems are equivalent, a solution to the original problem can be recovered from that of the new problem [38]. In particular, given a linear programming problem with d variables and n constraints, we can construct an equivalent linear programming problem for which the associated polyhedron has n variables, $d + n$ constraints and diameter bounded from above by $2(n - d)$, i.e., double the Hirsch bound. The thrust of the argument is that the diameter of the polyhedron associated with an arbitrary linear programming problem is not the only obstacle to the design of a strongly polynomial algorithm, because by manipulating the constraints, we can construct an equivalent linear programming problem for which a best possible sequence of pivots is small. The approach is in a similar spirit to the crisscross method and the arrangement method for the solution of linear programming problems, both of which traverse paths in the hyperplane arrangement – known to have diameter that that is quadratic in d and n – associated with a given linear programming problem [48, 34]. Hazan and Megiddo have shown that Koltun's arrangement method is equivalent to applying the simplex method to solve the so-called Phase 1 problem of finding a feasible solution for any linear programming problem [25], a fact which is implicit in Koltun's unpublished work. In other work of a similar style and purpose [18], De Loera and Onn showed that any linear programming problem in standard form is equivalent to a face of a 3-way transportation polytope. The graphs of 3-way transportation polytopes are known to have polynomial diameter due to the work of De Loera, Kim, Santos and Onn [19].

Theorem 6.1. *Given a linear programming problem Π_d defined by a d-dimensional cost vector $\mathbf{c}_d$ and polyhedron P_d with n facets, there exists an equivalent linear programming problem Π_n*



(a) The square $P^{(2)}$

(b) The cone $\mathbf{v}_3 P^{(2)}$

Figure 6.1: A simple polyhedron in $\mathbb{R}^2$ and cone in $\mathbb{R}^3$.

with the following properties:

(a) The defining cost vector $\mathbf{c}_n$ and polyhedron P_n can be constructed from the data of the given linear programming problem Π_d in linear time.

(b) The radius of P_n is no greater than $n - d$.

(c) There exists a monotone path from a starting vertex to an optimum vertex.

6.1 Example

We illustrate the ideas with reference to a 2D example. Consider a square $P^{(2)}$ in $\mathbb{R}^2$, as is illustrated in Figure 6.1a (a). For concreteness, suppose that the constraints are given by

$$\begin{cases} x_1 \leq 1 & (F_1) \\ x_2 \leq 1 & (F_2) \\ -x_1 \leq 1 & (F_3) \\ -x_2 \leq 1 & (F_4) \end{cases}.$$

Label the facets F_1, F_2, F_3, F_4. The diameter of $P^{(2)}$ clearly satisfies the Hirsch bound. That being said, we will proceed with the example in order to illustrate the approach.

Step 1. Given a k-dimensional polyhedron $P^{(k)}$, construct a cone $\mathbf{v}_{k+1}P^{(k)}$ in $\mathbb{R}^{k+1}$ with apex $\mathbf{v}_{k+1}$ and base $P^{(k)}$. The apex $\mathbf{v}_{k+1}$ may be any point in $\mathbb{R}^{k+1}$ lying outside the affine hull of $P^{(k)}$. For example, we construct the cone $\mathbf{v}_3P^{(2)}$ that is illustrated in Figure 6.1a (b) and defined by the constraints

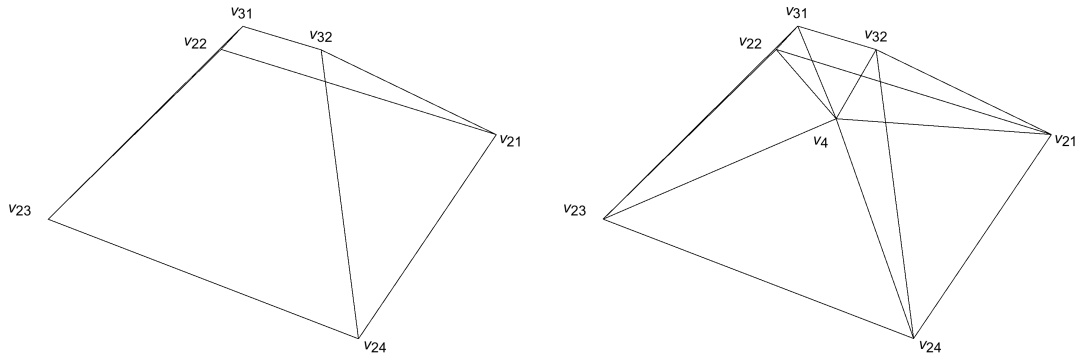
$$\begin{cases} x_1 + \frac{3}{4}x_3 \leq 1 & (F_1) \\ x_2 + \frac{3}{4}x_3 \leq 1 & (F_2) \\ -x_1 + \frac{3}{4}x_3 \leq 1 & (F_3) \\ -x_2 + \frac{3}{4}x_3 \leq 1 & (F_4) \\ -x_3 \leq 0 & (G_1) \end{cases}.$$

The lateral facets of the cone $\mathbf{v}_kP^{(k)}$ inherit their labels from the corresponding facets of the polyhedron $P^{(k)}$, whereas the "extra" facet is labelled G_{k-d}. Before proceeding we should point out that the cone has small diameter, i.e., diameter 2, and yet is not of much use as far as the simplex method is concerned. To see this, suppose that the simplex method reaches the apex of the cone, which is a highly degenerate vertex at which n lateral facets meet. In order to identify an improving edge, it would be necessary to consider up to $\binom{n}{d}$ combinations of lateral facets. So the problem of identifying an improving edge is as hard as the problem of identifying an optimum vertex for the given problem by computing all $\binom{n}{d}$ vertices of the associated hyperplane arrangement.

Step 2. Define inward and outward perturbations of a facet as follows. If a small rotation of the supporting hyperplane of a facet about one of its ridges results in a decrease in the Euclidean volume of the polyhedron, then the facet is said to have been perturbed *inward*. Otherwise, if the volume increases, then the facet is said to have been perturbed *outward*. The *foot* of a lateral facet of a cone $\mathbf{v}_{k+1}P^{(k)}$ is its intersection with the k-dimensional base $P^{(k)}$. Each perturbation to be performed is of a lateral facet of a cone either inward or outward about its foot. Such a perturbation can be achieved by adjusting the kth coordinate of the vector that is normal to the facet. By perturbing the lateral facets of the cone $\mathbf{v}_{k+1}P^{(k)}$ in this manner, we obtain a simple polyhedron $P^{(k+1)}$ containing $P^{(k)}$ as a k-dimensional face. For example, we may rotate the second and fourth facets of the cone $\mathbf{v}_3P^{(2)}$ inward to obtain a polyhedron $P^{(3)}$ that is illustrated in Figure 7.2 (b) and defined by the constraints

$$\begin{cases} x_1 + \frac{3}{4}x_3 \leq 1 & (F_1) \\ x_2 + x_3 \leq 1 & (F_2) \\ -x_1 + \frac{3}{4}x_3 \leq 1 & (F_3) \\ -x_2 + x_3 \leq 1 & (F_4) \\ -x_3 \leq 0 & (G_1) \end{cases}$$

We now perform a second iteration.



(a) The simple polyhedron $P^{(3)}$ (b) The cone $\mathbf{v}_4 P^{(3)}$

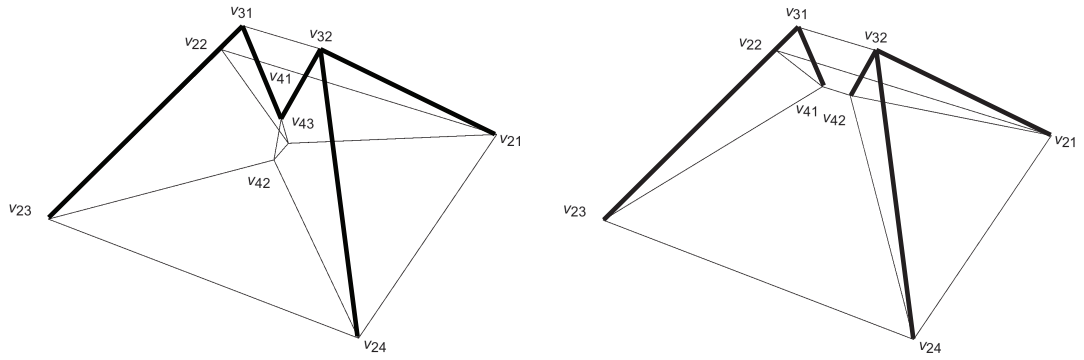
Figure 6.2: A simple polyhedron in $\mathbb{R}^3$ and cone in $\mathbb{R}^4$.

Step 1. Construct the cone $\mathbf{v}_4 P^{(3)}$ in $\mathbb{R}^4$ with apex $\mathbf{v}_4$ and base $P^{(3)}$. For example, we construct the cone $\mathbf{v}_4 P^{(3)}$ that is illustrated in Figure 6.2b and defined by the constraints

$$\begin{cases} x_1 + \frac{3}{4}x_3 + x_4 \leq 1 & (F_1) \\ x_2 + x_3 + \frac{3}{4}x_4 \leq 1 & (F_2) \\ -x_1 + \frac{3}{4}x_3 + x_4 \leq 1 & (F_3) \\ -x_2 + x_3 + \frac{3}{4}x_4 \leq 1 & (F_4) \\ -x_3 + x_4 \leq 0 & (G_1) \\ -x_4 \leq 0 & (G_2) \end{cases}$$

Step 2. The lateral facets of the cone $\mathbf{v}_4 P^{(3)}$ are labelled $\{F_1, F_2, F_3, F_4, G_1\}$. We can obtain a simple polyhedron by perturbing the extra facet G_1 inward or outward. For example, by perturbing the extra facet G_1 outward, we may obtain the polyhedron $P^{(4)}$ that is illustrated in Figure 6.3a and defined by the constraints

$$\begin{cases} x_1 + \frac{3}{4}x_3 + x_4 \leq 1 & (F_1) \\ x_2 + x_3 + \frac{3}{4}x_4 \leq 1 & (F_2) \\ -x_1 + \frac{3}{4}x_3 + x_4 \leq 1 & (F_3) \\ -x_2 + x_3 + \frac{3}{4}x_4 \leq 1 & (F_4) \\ -x_3 + \frac{1}{2}x_4 \leq 0 & (G_1) \\ -x_4 \leq 0 & (G_2) \end{cases}$$



(a) (b)

Figure 6.3: A polyhedron with (left) and without (right) a tree of radius $n - d$.

At this stage, it is worth pointing out that the goal of the approach is to construct a polyhedron $P^{(n)}$ that contains a tree $G(V, E)$, where the tree is of radius $n - d$ and interconnects the vertices of the original polyhedron $P^{(d)}$. With this goal in mind, we introduce some terminology. Consider the polyhedron $P^{(k)}$ in $\mathbb{R}^k$, where $d \leq k \leq n$, that we have obtained while following the approach. Let H_i denote the supporting hyperplane of the ith facet F_i of $P^{(k)}$. Let $Q^{(k)}$ denote the polyhedron in $\mathbb{R}^k$ that is defined by the supporting hyperplanes $H_1, H_2, \ldots, H_n$. In other words, we obtain $Q^{(k)}$ from $P^{(k)}$ by ignoring all the inequalities corresponding to extra facets. Define the vertex set $V^{(k)}$ as those vertices in $Q^{(k)}$ that are in $P^{(k)}$. Define the vertex set V as the disjoint union $\uplus_{k=d}^{n} V^{(k)}$. Finally, consider the edge set

$$E^{(k)} = \{(v, w) \in \text{edge}(P^{(k)}) | v \in V^{(k-1)}, w \in V^{(k)}\}, \qquad d < k \leq n.$$

Define the edge set E as the disjoint union $\uplus_{k=d+1}^{n} V^{(k)}$. Observe that if for all $d \leq k \leq n$, we have $\text{vert}(Q^{(k)}) \subseteq \text{vert}(P^{(k)})$, then the graph $G(V, E)$ is a tree. To see this, note that any vertex of $Q^{(k-1)}$, say, v is incident to exactly $k - 1$ facets of $Q^{(k)}$, which intersect to form a ray of $Q^{(k)}$. Thus, the vertex v is connected by a ray of $Q^{(k)}$ to one of its vertices, say, w. Moreover, if $v \in P^{(k-1)}$ and $w \in P^{(k)}$, then the edge set E contains the edge (v, w). It follows that if for all $d \leq k \leq n$, we have $\text{vert}(Q^{(k)}) \subseteq \text{vert}(P^{(k)})$, then the graph $G(V, E)$ is a tree.

If the vertex set $V^{(n)}$ is nonempty, then we call its only member $\hat{\mathbf{v}}$ the *root*. Clearly, if the graph $G(V, E)$ is a tree, then it has radius equal to $n - d$. For example, the tree $G(V, E)$ is marked with thick lines in Figure 5. Each vertex of $P^{(2)}$ is connected by a path of length 2 to the root $\hat{\mathbf{v}}$. The next example demonstrates that the graph $G(V, E)$ need not be connected to form a tree, in which case the above assertions cannot be made.

We can perturb the extra facet G_1 inward to obtain the polyhedron $P'^{(4)}$ that is illustrated in Figure 6.3b and defined by the constraints

$$\begin{cases} x_1 + \frac{3}{4}x_3 + x_4 \leq 1 & (F_1) \\ x_2 + x_3 + \frac{3}{4}x_4 \leq 1 & (F_2) \\ -x_1 + \frac{3}{4}x_3 + x_4 \leq 1 & (F_3) \\ -x_2 + x_3 + \frac{3}{4}x_4 \leq 1 & (F_4) \\ -x_3 + 2x_4 \leq 0 & (G_1) \\ -x_4 \leq 0 & (G_2) \end{cases}$$

In this example, it is not the case that for all $k = d, d+1, \ldots, n$, we have $\text{vert}(Q^{(k)}) \subseteq \text{vert}(P^{(k)})$. In particular, $\text{vert}(Q^{(4)}) \nsubseteq \text{vert}(P^{(4)})$. In other words, the polyhedron $P^{(4)}$ does not contain a root $\hat{\mathbf{v}}$. This example shows that it is important to carefully choose the perturbations. If at each iteration the extra facets are perturbed outward and "out of the way" of the points of intersection of the facets $F_1, F_2, \ldots, F_n$, then we can construct the desired tree $G(V, E)$ of radius $n - d$.

6.2 A general strategy

Now, given an arbitrary linear programming problem Π_d, we need a way to efficiently choose a set of perturbations of the facets, i.e., how to embed the given d-dimensional normal vectors in $\mathbb{R}^n$, for the approach to be of interest. First, it is computationally inexpensive to construct a cone. Supposing that $P^{(k)}$ is embedded in the hyperplane defined by $x_{k+1} = 0$ in $\mathbb{R}^{k+1}$, pick a point $\mathbf{v}^{(k+1)} = (v_1, v_2, \ldots, v_{k+1})$ such that $v_{k+1} > 0$ as the apex of the cone. For each lateral facet, determine the $(k+1)$th component of its defining normal vector $\mathbf{a} = (a_1, a_2, \ldots, a_{k+1})$ according to the linear equation

$$\mathbf{a} \cdot \mathbf{v}^{(k+1)} = b,$$

where the component b of the resource vector $\mathbf{b}$ is given in the problem data. Second, regarding the perturbations of the lateral facets of the cone corresponding to the facets of the original polyhedron, perhaps the easiest way to produce a simple polyhedron is by perturbing only one of the lateral facets inward. That is, increase by a small amount the $(k+1)$th component of the normal vector defining one of the facets $F_1, F_2, \ldots, F_k$. On the other hand, we need to perturb the extra facets "out of the way" of the points of intersection of the facets $F_1, F_2, \ldots, F_n$. Consider the normal vector $\mathbf{a}$ that defines an extra facet G of the polyhedron $P^{(k+1)}$. For a very large negative value of the component a_{k+1}, the dihedral angle between the extra facet G and the base $P^{(k)}$ is close to π and each vertex $\mathbf{v}$ of the polyhedron $Q^{(k+1)}$ satisfies the inequality $\mathbf{a} \cdot \mathbf{v} \leq b$. It is not necessary to set a numerical value for the component a_{k+1} of the normal vector defining the facet G. We can leave the value of a_{k+1} as an undetermined parameter, provided that we

modify the cost function $\mathbf{c} \cdot \mathbf{x}$.

The idea is to introduce a cost function of the form

$$\sum_{i=1}^{d} c_i x_i + \sum_{i=d+1}^{n} M_i x_i,$$

where $M_{d+1} \ll M_{d+2} \ll \ldots \ll M_n$. As a starting vertex of the polyhedron P_n, we choose the vertex $\hat{\mathbf{v}}$ that lies at the intersection of the lifted facets $F_1, F_2, \ldots, F_n$. For sufficiently large choices of $M_{d+1}, M_{d+2}, \ldots, M_n$, all the variables $x_{d+1}, x_{d+2}, \ldots, x_n$ are eventually driven to zero, which takes us back to the minimisation of the original cost function over the polyhedron $P^{(d)}$. Once again, there is no need to set numerical values for $M_{d+1}, M_{d+2}, \ldots, M_n$, which can be left as undetermined parameters [8]. We let the reduced costs be functions of the M_is, that is, whenever M_n is compared to another number (in order to determine whether a reduced cost is negative), M_n is treated as larger. Likewise, whenever M_{n-1} is compared to another number other than M_n, we treat M_{n-1} as larger, etc.

It is easy to see that there is a monotone path from the root to all vertices of $P^{(d)}$, including an optimum vertex. By construction, for all $d < k \leq n$, each vertex v of $Q^{(k-1)}$ is connected by an edge of the tree $G(V, E)$ to a vertex w of $Q^{(k)}$. Moreover, each such edge when directed from v to w makes an acute angle with the kth unit vector $\mathbf{e}_k$ while being orthogonal to the unit vectors $\mathbf{e}_{k+1}, \mathbf{e}_{k+2}, \ldots, \mathbf{e}_n$. It follows that the path from any vertex of the original polyhedron $P^{(d)}$ toward the root $\hat{\mathbf{v}}$ is monotonically increasing.

To conclude this chapter, we note that the lifted polyhedron $P^{(n)}$ has the same dimension and number of facets as the polyhedron associated with the auxiliary problem for which the objective is to find a feasible solution of the given linear programming problem. In particular, the polyhedron associated with the auxiliary problem is the intersection of the nonnegative orthant in $\mathbb{R}^n$ with d half spaces. The facets of the nonnegative orthant correspond to the facets $F_1, F_2, \ldots, F_n$ of the polyhedron associated with the given linear program while the d half spaces correspond to the extra facets as in our procedure. The difference between the polyhedron associated with the auxiliary linear programming problem and the polyhedron that we have constructed is that the former is not guaranteed to contain a tree of radius $n - d$ interconnecting the basic feasible solutions.

Appendix A

Appendix

A GEOMETRIC APPROACH TO SHORTEST BOUNDED CURVATURE PATHS

JOSÉ AYALA, DAVID KIRSZENBLAT, AND HYAM RUBINSTEIN

Abstract. We present a geometric proof for the classification of length minimisers in spaces of planar bounded curvature paths. Our methods can be adapted without much effort to classify length minimisers in spaces of bounded curvature paths in other surfaces. The main result in this note fills a gap by furnishing a geometric proof for a problem in geometry that hither to was missing from the literature.

1. Motivation

In this work we present a constructive proof for the classification of global length minimisers in spaces of planar bounded curvature paths first obtained by L. Dubins in [14]. Our approach is flexible and easy to generalise. In particular, the techniques developed in this note allowed us to achieve:

- the classification of the homotopy classes of bounded curvature paths [5];
- the classification of length minimisers in homotopy classes of bounded curvature paths [3].

In general terms, a planar bounded curvature path corresponds to a C^1 and piecewise C^2 path lying in $\mathbb{R}^2$. These paths connect two elements in the tangent bundle $T\mathbb{R}^2$ and have curvature bounded by a positive constant κ. As proved by L. Dubins in [14] a C^1 path of minimal length lying in $\mathbb{R}^2$, having its curvature bounded by a positive constant, connecting two elements of the tangent bundle $T\mathbb{R}^2$ is indeed a piecewise C^2 path (this is the well known CSC, CCC characterisation [14]). The view put forward by L. Dubins regarding the bound on the curvature (an interpretation followed by many) comes from considering a particle that moves at constant speed subject to a maximum possible force. Another physical interpretation can be the following: a uniform bending beam has an internal structure which depends on certain binding forces; when exposed to tension the beam may be deformed - such deformations always have a limit given by the beam's material resistance. Length minimising bounded curvature paths, widely known as Dubins paths, have proven to be extraordinarily useful in applications since a bound on the curvature is a turning circle constraint for the trajectory of a vehicle along a path. In contrast, our approach deals with curves as topological objects.

To characterise the length minimisers we proceed as follows. Start with an arbitrary bounded curvature path and consider a partition of it so that the length of each piece is less than $\frac{1}{\kappa}$. Replace each piece of the path, called a *fragment*, by a piecewise constant curvature path, called a *replacement*, making sure the

2000 *Mathematics Subject Classification.* Primary 49Q10; Secondary 90C47, 51E99, 68R99.
Key words and phrases. Bounded curvature paths, Dubins paths, path optimisation.

length of each replacement is at most the length of the respective fragment to be replaced. As a consequence of the previously described process, we obtain a bounded curvature path corresponding to a finite number of concatenated piecewise constant curvature paths (a *cs* path) called a *normalisation*. Notice that there is an implicit sense of continuity when replacing a bounded curvature path by a normalisation since fragments can be chosen to be arbitrarily small. However, a continuity argument preserving the bound on curvature is only achieved in [5]. After obtaining a normalisation for a given path, the key idea to construct a length minimising path, is to study the path *complexity*, which is the number of constant curvature components in a *cs* path. In particular, we develop a series of results showing that if a *cs* path has at least 4 components, a replacement can be made of a portion of the path to reduce both the path complexity and the length. This process terminates with a length minimising *cs* path of complexity at most 3. There is also a homotopy involved in this process but no claim about continuity is needed.

In 1887 A. Markov in [26] initiated this theory by studying the optimal linking of railroad tracks. Seventy years later L. Dubins in [14, 15] obtained the first general results in this theory. In between the work of L. Dubins and A. Markov there are a number of scattered results due to V. Ionin, G. Pestov, E. Schmidt, A. Schur and H. Schwarz [7, 30, 32]. There are a number of proofs for the characterisation of length minimisers, each of them with a different flavour. In 1974 J. Johnson in [20] characterised length minimisers by applying the Pontryagin maximum principle. Later in 1990 Reeds and Shepp studied the length minimisers for the trajectories of a car moving forward and backward. Recently S. Eriksson-Bique et. al. in [18] introduced a discrete analogue to Dubins paths. In [28] F. Monroy-Pérez also applied the Pontryagin maximum principle to obtain the length minimisers in 2-dimensional homogeneous spaces of constant curvature. Our proof is elementary, easy to implement and provides a foundation for a continuity argument for the deformation of bounded curvature paths [5]. This step permits us to push the theory forward outside optimality. In addition, our techniques establish an elementary framework to study bounded curvature paths in other surfaces.

Bounded curvature paths have proven to be useful in science and engineering as well as in mathematics. These paths have been extensively studied in computer science [1, 9, 18, 24, 33], control theory [21, 27, 28, 29, 34, 35, 36], engineering [2, 8, 11, 13, 23] and recently in geometric knot theory and the Steiner tree problem [10, 16, 17, 22]. It would be interesting to study the computational complexity of reducing a *cs* path to the length minimiser given a set of operations on *cs* paths reducing the length and complexity. Currently the first author is implementing the techniques presented in this work in Dubins Explorer, a software for bounded curvature paths [12]. In general, a bound on curvature is a property widely observed in nature: in potamology, in the formation of meanders [25] and in the geometry of coral structures [19].

2. Normalising Bounded Curvature Paths

We start this work by presenting basic definitions. Then, we procede to introduce the concept of normalisation for bounded curvature paths. The idea is to (after some point) only deal with bounded curvature paths corresponding to a finite number of concatenations of line segments and arcs of unit circles. Let us denote by $T\mathbb{R}^2$ the tangent bundle of $\mathbb{R}^2$. The elements in $T\mathbb{R}^2$ correspond to pairs (x, X) sometimes

denoted just by X. As usual, the first coordinate corresponds to a point in $\mathbb{R}^2$ and the second to a tangent vector in $\mathbb{R}^2$ at x.

Definition 2.1. Given $(x, X), (y, Y) \in T\mathbb{R}^2$, we say that a path $\gamma : [0, s] \to \mathbb{R}^2$ connecting these points is a *bounded curvature path* if:

- γ is C^1 and piecewise C^2.
- γ is parametrized by arc length (i.e. $||\gamma'(t)|| = 1$ for all $t \in [0, s]$).
- $\gamma(0) = x$, $\gamma'(0) = X$; $\gamma(s) = y$, $\gamma'(s) = Y$.
- $||\gamma''(t)|| \leq \kappa$, for all $t \in [0, s]$ when defined, $\kappa > 0$ a constant.

Of course, s is the arc-length of γ.

The first condition means that a bounded curvature path has continuous first derivative and piecewise continuous second derivative. For the third condition to make sense, without loss of generality, we extend the domain of γ to $(-\epsilon, s + \epsilon)$ for $\epsilon > 0$. The third item is called the endpoint condition. The bound on the curvature $\kappa > 0$ can be chosen to be $\kappa = 1$ by considering a suitable scaling of the plane. Generally, the interval $[0, s]$ is denoted by I. Denote by $\mathcal{L}(\gamma)$ the length of γ and $\mathcal{L}(\gamma, a, b)$ the length of γ restricted to $[a, b] \subset I$.

Definition 2.2. Given $\mathrm{X}, \mathrm{Y} \in T\mathbb{R}^2$. The space of bounded curvature paths starting at x tangent to X finishing at y tangent to Y is denoted by $\Gamma(\mathrm{X},\mathrm{Y})$.

Important properties of $\Gamma(\mathrm{X},\mathrm{Y})$ (e.g., the kind of length minimiser, the number of length minimisers, the number of connected components) depend on the endpoint condition. Also, the existence of a homotopy class in $\Gamma(\mathrm{X},\mathrm{Y})$ whose elements are embedded, depends on the chosen endpoint [4, 5].

Consider the origin of an orthogonal coordinate system in $\mathbb{R}^2$ as the base point x with X lying in the abscissa. Note that the arc-length parametrisation implies the endpoints are elements in the unit tangent bundle $UT\mathbb{R}^2$. This space is equipped with a natural projection $\pi : UT\mathbb{R}^2 \to \mathbb{R}^2$. The fiber $\pi^{-1}(x)$ corresponds to $\mathbb{S}^1$ for all $x \in \mathbb{R}^2$. The space of endpoint conditions corresponds to a sphere bundle on $\mathbb{R}^2$ with fiber $\mathbb{S}^1$.

Observe that in a bounded curvature path the osculating circle at each point (when defined) has radius at least 1. In Figure 1 we illustrate some bounded curvature paths and osculating circles.

Definition 2.3. Let $\mathrm{C}_l(\mathrm{X})$ be the unit circle tangent to x and to the left of X. An analogous interpretation applies for $\mathrm{C}_r(\mathrm{X})$, $\mathrm{C}_l(\mathrm{Y})$ and $\mathrm{C}_r(\mathrm{Y})$ (see Figure 1). These circles are called *adjacent circles*. We denote their centers with lower-case letters, so the center of $\mathrm{C}_l(\mathrm{X})$ is denoted by $c_l(\mathrm{X})$.

Definition 2.4. A *cs* path is a bounded curvature path corresponding to a finite number of concatenations of line segments (denoted by S) and arcs of a unit circle (denoted by C) see Figure 1. Let R denote a clockwise traversed arc and L a counterclockwise traversed arc. The line segments and arcs of circles are called *components*. The number of components is called the *complexity* of the path.

Definition 2.5. A *fragmentation* of a bounded curvature path $\gamma : I \to \mathbb{R}^2$ corresponds to a finite sequence $0 = t_0 < t_1 \ldots < t_m = s$ of elements in I such that, $\mathcal{L}(\gamma, t_{i-1}, t_i) < 1$ with $\sum_{i=1}^{m} \mathcal{L}(\gamma, t_{i-1}, t_i) = s$. We denote by a *fragment*, the restriction of γ to the interval determined by two consecutive elements in the fragmentation.

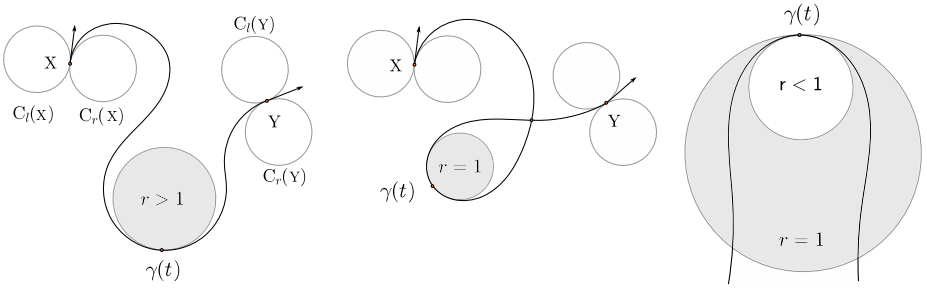


FIGURE 1. Left and centre: Examples of bounded curvature paths and adjacent circles. Right: An example when the curvature bound is violated. Note that the radius of the osculating circle $r < 1$ at $\gamma(t)$ implies $\kappa > 1$.

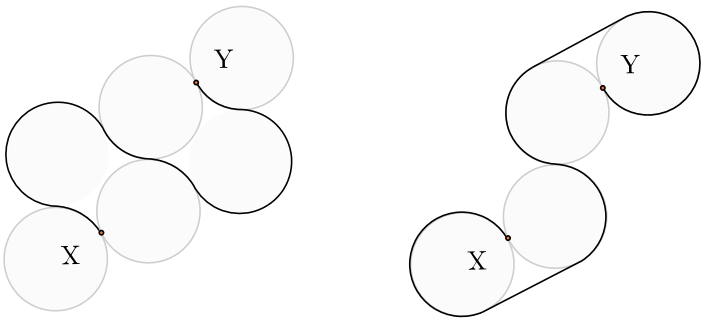


FIGURE 2. Examples of *cs* paths in $\Gamma(\mathrm{x},\mathrm{y})$.

Definition 2.6. Let $\mathrm{z} \in T\mathbb{R}^2$ with $\mathrm{z} = (z, Z)$, where the first component corresponds to the origin of a coordinate system with abscissa x and ordinate y, and $Z = (1, 0)$. Denote by $\mathcal{R}(\mathrm{z})$ the region enclosed by the complement of the union of the interior of the disks with boundary $\mathrm{C}_l(\mathrm{z})$ and $\mathrm{C}_r(\mathrm{z})$ intersected with a unit radius disk centered at z denoted by D_z. Let $\mathcal{R}^+(\mathrm{z})$ be the set of points in $\mathcal{R}(\mathrm{z})$ with $x > 0$ and $\mathcal{R}^-(\mathrm{z})$ be the set of points in $\mathcal{R}(\mathrm{z})$ with $x < 0$ (see Figure 3).

The next proposition uses a technical result. We propose that the reader refer to Corollary 2.4 in [**?**] for details. The omission of such a result should not compromise the reader's understanding. Corollary 2.4 in [**?**] states that a bounded curvature path making a u-turn must have length at least 2 (see dashed trace in Figure 3).

Proposition 2.7. Given $\mathrm{z} \in T\mathbb{R}^2$. A fragment $\gamma : I \to \mathcal{R}(\mathrm{z})$ not entirely lying in $\partial\mathcal{R}(\mathrm{z})$ with $\gamma(t) = z$ and $\gamma'(t) = Z$ does not intersect $\partial\mathcal{R}^-(\mathrm{z})$ or $\partial\mathcal{R}^+(\mathrm{z})$ (see Figure 3 right).

Proof. Choose $\mathrm{z} \in T\mathbb{R}^2$ and suppose that a fragment γ is contained in $\mathcal{R}(\mathrm{z})$ with $\gamma(t) = z$ and $\gamma'(t) = Z$. Without loss of generality suppose γ intersects $\partial^+\mathcal{R}(\mathrm{z})$ in $\mathrm{C}_r(\mathrm{z})$ at $\gamma(t')$ for some $t' \in I$. By considering $z = p$ and $\gamma(t') = q$ as in Corollary 2.4 in [**?**] we obtain that the length of γ is at least 2. Since the diameter of $\partial\mathcal{R}^+(\mathrm{z})$

is equal to 1 we conclude the proof. The other cases are proved by applying a similar argument. □

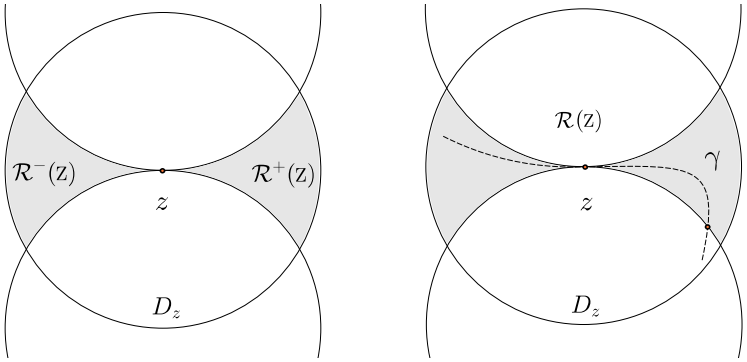


FIGURE 3. Left: The grey region corresponds to $\mathcal{R}(\mathrm{z})$. Right: The dashed trace is not a fragment since by Proposition 2.7 a fragment never leaves $\mathcal{R}(\mathrm{z})$ through $\mathrm{C}_l(\mathrm{z})$ or $\mathrm{C}_r(\mathrm{z})$.

We include the proofs for the following simple but crucial lemmata. These give lower bounds for the lengths of curves when compared with arcs of unit circles and line segments. Consider a curve $\gamma(t) = (r(t)\cos\theta(t), r(t)\sin\theta(t))$ in polar coordinates.

Lemma 2.8. For any curve $\gamma : [0, s] \to \mathbb{R}^2$ with $\gamma(0) = (1, 0)$, $r(t) \geq 1$, and $\theta(s) = \eta$, one has $\mathcal{L}(\gamma) \geq \eta$ (see Figure 4 left).

Proof. Consider γ as defined above with $r, \theta : [0, s] \to \mathbb{R}$. Then we immediately have that:

$$\mathcal{L}(\gamma) = \int_0^s |\gamma'(t)|\, dt = \int_0^s \sqrt{|r'(t)|^2 + r(t)^2\,|\theta'(t)|^2}\; dt \geq \int_0^s |\theta'(t)|\; dt \geq \eta$$

□

Lemma 2.9. For any C^1 curve $\gamma : [0, s] \to \mathbb{R}^2$ with $\gamma(0) = (0, 0)$, $\gamma(s) = (x, z)$ and $z \geq 0$, one has $\mathcal{L}(\gamma) \geq z$ (see Figure 4 right).

Proof. Consider $\gamma(t) = (x(t), y(t))$ with $x, y : [0, s] \to \mathbb{R}$. Then,

$$\mathcal{L}(\gamma) = \int_0^s |\gamma'(t)|\, dt = \int_0^s \sqrt{x'(t)^2 + y'(t)^2}\; dt \geq \int_0^s |y'(t)|\; dt \geq z$$

□

Definition 2.10. A bounded curvature path $\gamma : I \to \mathbb{R}^2$ in $\Gamma(\mathrm{x},\mathrm{y})$ has a negative direction if there exists $t \in I$ such that $\langle X, \gamma'(t)\rangle < 0$.

Corollary 2.11. A fragment does not have a negative direction.

Proof. It is easy to see from Lemma 2.8 that a negative direction implies the length of a bounded curvature path is greater than $\frac{\pi}{2}$. Since fragments have length less than 1 the result follows. □

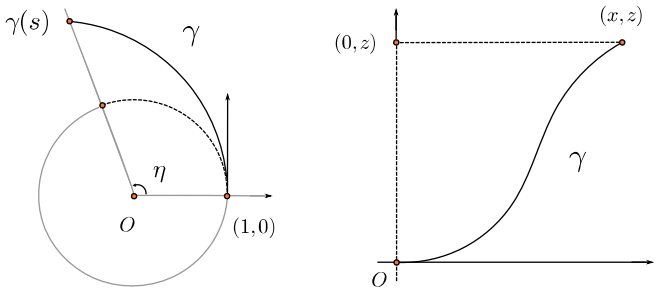


FIGURE 4. Left: Illustration of Lemma 2.8. Right: Illustration of Lemma 2.9.

In [4] we proved that for certain endpoints in $T\mathbb{R}^2$ there exists a compact region $\Omega \subset \mathbb{R}^2$ that *traps* embedded bounded curvature paths (see Figure 5 right). That is, no embedded bounded curvature lying in Ω can be deformed while preserving the curvature bound to a path having a point in the complement of Ω.

Proposition 2.12. A fragment γ such that $\gamma(t) = z$ and $\gamma'(t) = Z$ is contained in $\mathcal{R}(\mathrm{Z})$.

Proof. Suppose a fragment γ intersects the boundary of the unit disk D_z at $P = \gamma(t)$ for some $t \in I$. By virtue of Lemma 2.9 the length of γ is greater than or equal to the length of $\overline{zP} = 1$ leading to a contradiction (see Figure 5 left). By Proposition 2.7 fragments inside $\mathcal{R}(\mathrm{Z})$ that are not arcs of an adjacent circle do not intersect $\partial\mathcal{R}(\mathrm{Z})$. We conclude that fragments are confined in $\mathcal{R}(\mathrm{Z})$. □

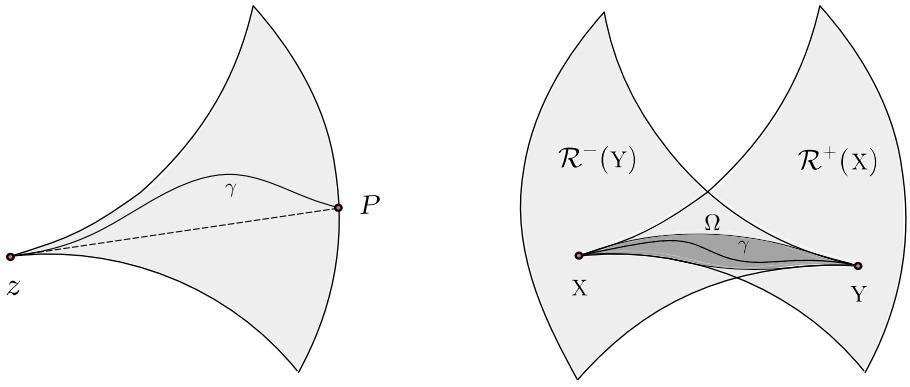


FIGURE 5. Left: Applying Lemma 2.9 to γ in $\mathcal{R}^+(\mathrm{Z})$. Right: From [4] we conclude that a fragment and replacement are trapped in a compact subset $\Omega \subset \mathcal{R}^+(\mathrm{X}) \cap \mathcal{R}^-(\mathrm{Y})$.

The definition of a *cs* path is such that the points where the curvature is not defined are precisely those between concatenated C and S components. In a CSC path call these points P and Q (see Figure 6).

Proposition 2.13. For a fragment in $\Gamma(\mathrm{X},\mathrm{Y})$ there exists a CSC path in $\Gamma(\mathrm{X},\mathrm{Y})$ having circular components of length less than π.

Proof. Let γ be a fragment with endpoint condition $\mathrm{X},\mathrm{Y} \in T\mathbb{R}^2$. By Corollary 2.11 fragments do not have a negative direction, therefore, there is a *unique* line segment from x tangent to $\mathrm{C}_l(\mathrm{Y})$ or $\mathrm{C}_r(\mathrm{Y})$. Consider the segment $\overline{xy}$. We can have three scenarios: $\overline{xy}$ crosses $\mathrm{C}_l(\mathrm{X})$, $\overline{xy}$ crosses $\mathrm{C}_r(\mathrm{X})$ or $\overline{xy}$ is tangent to $\mathrm{C}_l(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{X})$. Symmetrically we can have: $\overline{yx}$ crosses $\mathrm{C}_l(\mathrm{Y})$, $\overline{yx}$ crosses $\mathrm{C}_r(\mathrm{Y})$ or $\overline{yx}$ is tangent to $\mathrm{C}_l(\mathrm{Y})$ and $\mathrm{C}_r(\mathrm{Y})$. First, suppose the segment $\overline{xy}$ crosses both $\mathrm{C}_r(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{Y})$. Observe that there is a common tangent line to $\mathrm{C}_r(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{Y})$ with respective tangent points say P and Q. Also, there are arcs in $\mathrm{C}_r(\mathrm{X})$, $\mathrm{C}_r(\mathrm{Y})$ from x to P and Q to y traversed counterclockwise. We define $\beta : I \to \mathbb{R}^2$ to be the arc-length parameterized CSC path of type RSR if the segment $\overline{xy}$ crosses both $\mathrm{C}_r(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{Y})$ by considering the common tangent segment $\overline{PQ}$ to $\mathrm{C}_r(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{Y})$ and the arc from x to P and the arc from Q to y (see Figure 6 right). In this fashion, β corresponds to a bounded curvature path that starts from x, travels along $\mathrm{C}_r(\mathrm{X})$ until reaching P, then travels along $\overline{PQ}$ and then travels along $\mathrm{C}_r(\mathrm{Y})$ from Q until reaching y. Similar reasoning applies for the construction of LSR, LSL, RSL path types under the other possible intersections of $\overline{xy}$ with the adjacent circles. By Corollary 2.11 fragments do not have a negative direction and by Lemma 2.8 the replacement has circular arcs of length less than π. □

The following key result is proved by a direct projection argument. The idea is to divide up a fragment into at most three pieces and then project such pieces onto the replacement constructed in Proposition 2.13. We conclude that the replacement is never longer than the fragment.

Lemma 2.14. *The length of a replacement is at most the length of the associated fragment with equality if and only if these paths are identical (see Figure 6).*

Proof. Consider a fragment γ with endpoint condition $\mathrm{X},\mathrm{Y} \in T\mathbb{R}^2$. Without loss of generality consider $x = (0,0)$ and $X = (1,0)$. Construct a replacement path β as in Proposition 2.13 and suppose that $\gamma \neq \beta$. Denote by L_1 the line passing through the center of the first adjacent arc in β and the common point P between the first and the second component of β and denote by L_2 the line passing through the center of the second arc of β and the common point Q between the second and the third component of β. Observe that L_1 and L_2 are parallel lines. By virtue of Proposition 2.12 we have that γ must lie in $\mathcal{R}^+(\mathrm{X})$ and in particular γ lies outside the interior of the adjacent circles associated with X and Y. By continuity, the path γ must cross the lines L_1 and L_2 at, say, the points O and N respectively (see Figure 6). Denote by γ_1 the portion of γ in between x and the first time γ intersects L_1; denote by γ_2 the portion of γ in between the first time γ intersects L_1 and the first time γ intersects L_2 and denote by γ_3 the portion of γ in between the first time γ intersects L_2 and y. Suppose $\gamma \neq \beta$ at some point in γ_1. By Proposition 2.12 the fragment γ does not intersect $\partial\mathcal{R}^+(\mathrm{X})$ and therefore $P \neq O$ (also $N \neq Q$). By virtue of Lemma 2.8 the length of γ_1 is greater than the length of β (the angle traveled in the first component of β from x to P). By applying Lemma 2.9 to γ_2 we conclude that the length of γ in between O and N is greater than the length of the segment PQ. By applying Lemma 2.8 (as we did for γ_1) we have that the length of γ_3 is greater than or equal to the length of β in between Q and y. Therefore,

γ must be longer than β. The cases $\gamma \neq \beta$ at some point in γ_2 or γ_3 are proven identically as above. □

Notice that fragments have length less than 1. Such a bound is sufficient to guarantee that the replacement is a CSC path possibly with some components of zero length. On the other hand, CSC paths can be sometimes constructed for longer pieces in a bounded curvature path.

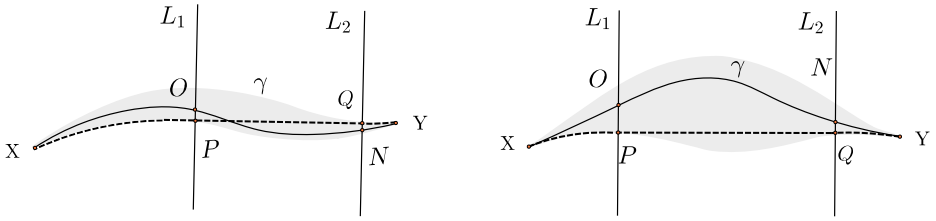


FIGURE 6. The points P and Q are the points in the dashed CSC path where the curvature is not defined. The dashed trace corresponds to the replacement β constructed in Lemma 2.14. The grey regions correspond to Ω.

Proposition 2.15. A path $\gamma \in \Gamma(\mathrm{X},\mathrm{Y})$ may be replaced by a *cs* path of length at most the length of γ.

Proof. Consider a fragmentation for $\gamma \in \Gamma(\mathrm{X},\mathrm{Y})$. By applying Lemma 2.14 to each fragment we obtain a bounded curvature path of length at most the length of γ being a finite number of concatenated CSC paths. □

3. Length Minimising Bounded Curvature Paths

Here we continue the process of reducing bounded curvature paths in terms of length and complexity by looking at larger pieces in a *cs* paths. We conclude in particular that length minimising bounded curvature paths have complexity at most three.

Definition 3.1. A component of type $\mathcal{C}_1$ is a CSCSC path as shown in Figure 7 left and center. A component of type $\mathcal{C}_2$ is a CSCCSC path as shown in Figure 7 right.

Definition 3.2. Let $\gamma \in \Gamma(\mathrm{X},\mathrm{Y})$ be a component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$). Then γ is called *admissible* if a replacement can be constructed in $\Gamma(\mathrm{X},\mathrm{Y})$. A component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$) that is not admissible is called a non-admissible component (see Figure 7).

Proposition 3.3. Given $\mathrm{X},\mathrm{Y} \in T\mathbb{R}^2$. A *cs* path in $\Gamma(\mathrm{X},\mathrm{Y})$ containing an admissible component as a sub path can be replaced by another *cs* path in $\Gamma(\mathrm{X},\mathrm{Y})$ with less complexity and with the length of the latter being at most the length of the former.

Proof. Consider a *cs* path containing an admissible component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$). By applying the same construction in Lemma 2.14 to the *cs* path in between the component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$) (see Figure 8 left) the length of the replacement is seen to be at most the length of the component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$) concluding the proof. □

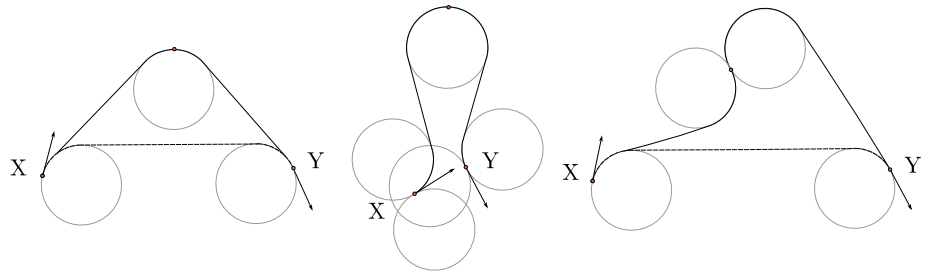


FIGURE 7. Examples of components of type $\mathcal{C}_1$ and $\mathcal{C}_2$. The dashed traces in the left and right figures are replacements. The middle illustration corresponds to a non-admissible component of type $\mathcal{C}_1$. Note that no replacement can be constructed for such a component.

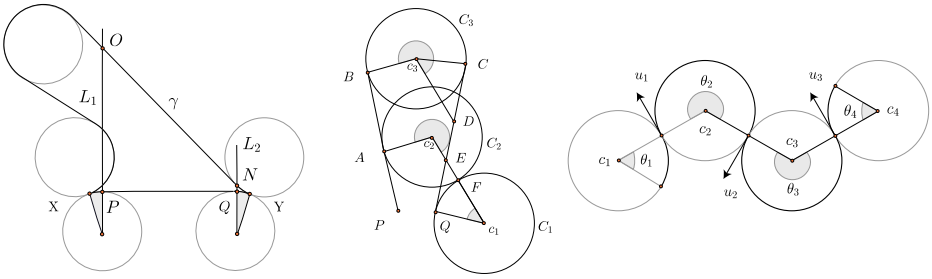


FIGURE 8. Left: The restriction of a *cs* path γ to an admissible component of type $\mathcal{C}_1$. Lemma 2.14 is trivially adapted to a component of type $\mathcal{C}_1$ (or $\mathcal{C}_2$) (see notation in Figure 6). Center: The notation in Proposition 3.4. The path γ is the SCS path from P to Q. Right: The notation for the non optimality of CCCC paths.

Theorem 3.4. *Components of type $\mathcal{C}_1$ and $\mathcal{C}_2$ are not length minimisers.*

Proof. Consider a component of type $\mathcal{C}_1$. If the component is admissible then by applying Proposition 3.3 the result follows. If the component of type $\mathcal{C}_1$ is non-admissible then consider the SCS part of the component of type $\mathcal{C}_1$, denote it by γ, and without loss of generality suppose the line segments in γ have the same length. Referring to Figure 8 (center) for notation, we proceed to construct a path shorter than γ. Note that γ is the SCS path from P to Q with the length of $\overline{PB}$ and $\overline{CQ}$ being the same. Denote by C_1 the left adjacent circle at Q. Let C_2 be the circle that is simultaneously tangent to C_1 and $\overline{PB}$. Let C_3 be the circle containing the middle component of γ; denote the centers of these circles in lowercase. The parallel line to $\overline{c_1\,c_2}$ passing through c_3 intersects $\overline{CQ}$ at D. The segment $\overline{c_1\,c_2}$ intersects $\overline{CQ}$ at E. Since C_3 is above C_2 (both circles being tangent to $\overline{PB}$) then by applying Lemma 2.8 we conclude that the length of γ between the points B and D is greater than the length of the arc in C_2 between A and F. Again by Lemma 2.8 we conclude that the length of the shorter arc between Q and F in C_1 is less

than the length of the segment $\overline{QE}$. Denote by δ the SCC path from P to Q with first component $\overline{PA}$, second component the arc AF in C_2 and third component the arc FQ lying in C_1. We conclude that $\mathcal{L}(\delta) < \mathcal{L}(\gamma)$ implying that the component of type $\mathcal{C}_1$ is not a length minimiser. If a component of type $\mathcal{C}_2$ is admissible then by applying Lemma 2.8 the result follows. If a component of type $\mathcal{C}_2$ is non-admissible, then by applying an identical argument as in the previous paragraphs the result follows. □

Corollary 3.5. The SCS and CCS (or SCC) paths are not length minimisers.

Proof. By the construction in Theorem 3.4 we have that SCS paths are not length minimisers. The proof that CCS (or SCC) paths are also not length minimisers follows from an identical construction as the one used for the non-admissible case in Theorem 3.4. We leave the details to the reader. □

Proposition 3.6. A CCC path with middle arc of length less than or equal to π is not a length minimiser.

Proof. Without loss of generality consider an LRL path. An LRL path with middle component of length less than or equal to π has the center of the circle containing R below the line joining $c_l(\mathrm{X})$ and $c_l(\mathrm{Y})$. Consider the adjacent circles $\mathrm{C}_l(\mathrm{X})$ and $\mathrm{C}_l(\mathrm{Y})$ and construct a replacement as in Proposition 2.13. Apply Lemma 2.14 to the RLR path to conclude the statement. If we consider an RLR path with middle component of length less than or equal to π then consider $\mathrm{C}_r(\mathrm{X})$ and $\mathrm{C}_r(\mathrm{Y})$ and proceed as before. □

Proposition 3.7. The *cs* paths of complexity four are not length minimisers.

Proof. If a complexity four path contains exactly one line segment then it must contain a CCS or SCC component. By virtue of Corollary 3.5 we have that such a component is not a length minimiser. If a complexity four path contains exactly two line segments then it must contain an SCS component. Again by Corollary 3.5 we have that such a component is not a length minimiser.

For a proof that CCCC paths are not length minimisers, we use a variational argument. In particular, we show that if a CCCC path is a critical point of the length function, then it is unstable. So that it is evident how the equations scale as the minimum turning radius κ varies, we do not set κ equal to unity. Let γ be a CCCC path whose arcs have lengths $\theta_1(t), \theta_2(t), \theta_3(t)$ and $\theta_4(t)$ and centres $c_1(t), c_2(t), c_3(t)$ and $c_4(t)$, respectively. For succinctness, we suppress the dependency on t. Hence the length of γ is given by

$$\mathcal{L}(\gamma) = \theta_1 + \theta_2 + \theta_3 + \theta_4.$$

Consider a perturbation of γ to a nearby path of type CCCC. The first variation of length is given by

$$\dot{\mathcal{L}}(\gamma) = \langle (c_2 - c_1)', \hat{u}_1 \rangle + \langle (c_3 - c_2)', \hat{u}_2 \rangle + \langle (c_4 - c_3)', \hat{u}_3 \rangle,$$

where the unit vectors $\hat{u}_i$ are tangent to the circular arcs of γ with centres c_i and c_{i+1} and have the same orientation as γ (see Figure 8 right). Note that the lengths $\|c_{i+1} - c_i\|$ remain constant throughout the perturbation. Hence, the vector $(c_{i+1} - c_i)'$ must be perpendicular to the vector $c_{i+1} - c_i$. In other words, the vector $(c_{i+1} - c_i)'$ must be parallel to the vector $\hat{u}_i$. So we obtain $\langle (c_{i+1} - c_i)', \hat{u}_i \rangle = \pm \|(c_{i+1} - c_i)'\|$, where the sign of the inner product depends on the direction of

variation. In particular, the inner products $\langle (c_2 - c_1)', \hat{u}_1 \rangle$ and $\langle (c_4 - c_3)', \hat{u}_3 \rangle$ are of opposite sign. On the other hand, the inner product $\langle (c_3 - c_2)', \hat{u}_2 \rangle$ may change sign relative to $\langle (c_2 - c_1)', \hat{u}_1 \rangle$ and $\langle (c_4 - c_3)', \hat{u}_3 \rangle$. Note that the centres c_1 and c_4 are fixed. Hence both $c_1' = 0$ and $c_4' = 0$. At the critical point, we set the derivative of the length function $\mathcal{L}(\gamma)$ equal to zero and obtain one of the following two equations:

$$\dot{\mathcal{L}}(\gamma) = \|c_2'\| + \|c_3' - c_2'\| - \|c_3'\| = 0,$$

or

$$\dot{\mathcal{L}}(\gamma) = \|c_2'\| - \|c_3' - c_2'\| - \|c_3'\| = 0.$$

In either case, the vectors c_2' and c_3' must be parallel by the triangle inequality. Since the vector $c_3 - c_2$ cannot undergo any change in length, the perturbation vectors c_2' and c_3' must in fact be equal. Note that the perturbation vectors c_2' and c_3' are tangent to the circles of radius 2κ with centres c_1 and c_4. It is geometrically obvious that a stationary configuration is obtained only in the symmetric situation where $\hat{u}_1 = \hat{u}_3$. In order to demonstrate that this configuration corresponds to an unstable critical point of the length function, we will need to study the second variation of length. By the Leibniz rule,

$$\ddot{\mathcal{L}}(\gamma) = \langle (c_2 - c_1)'', \hat{u}_1 \rangle + \langle (c_2 - c_1)', \hat{u}_1' \rangle + \langle (c_3 - c_2)'', \hat{u}_2 \rangle + \langle (c_3 - c_2)', \hat{u}_2' \rangle + \langle (c_4 - c_3)'', \hat{u}_3 \rangle + \langle (c_4 - c_3)', \hat{u}_3' \rangle.$$

By the same reasoning as above, the vector $(c_{i+1} - c_i)'$ must be perpendicular to the vector $\hat{u}_i'$, and both $c_1'' = 0$ and $c_4'' = 0$. Moreover, at the critical point, $\hat{u}_1 = \hat{u}_3$. Therefore,

$$\ddot{\mathcal{L}}(\gamma) = \langle (c_3 - c_2)'', \hat{u}_2 \rangle - \langle (c_3 - c_2)'', \hat{u}_1 \rangle.$$

The vector $(c_3 - c_2)''$ makes an obtuse angle with the vector $\hat{u}_2$ throughout the perturbation and is parallel to the vector $\hat{u}_2$ at the critical point. To see this, note that the component of $(c_3 - c_2)''$ parallel to $c_3 - c_2$ is equal to $-|\langle (c_3 - c_2)', \hat{u}_2 \rangle|^2 / 2\kappa$, and akin to the centripetal acceleration of the vector $c_3 - c_2$. At the critical point, $c_2' = c_3'$. Hence, the vector $c_3 - c_2$ is not subject to any rotation and the component of $(c_3 - c_2)''$ perpendicular to $c_3 - c_2$ must be zero. Therefore,

$$\ddot{\mathcal{L}}(\gamma) = -\|(c_3 - c_2)''\| - \langle (c_3 - c_2)'', \hat{u}_1 \rangle < 0,$$

and the configuration is unstable. □

Corollary 3.8. *cs* paths of complexity greater than 3 are not length minimisers.

Proof. Immediate from Proposition 3.7. □

Now we proceed to establish the classification of length minimisers in spaces of planar bounded curvature first obtained in [14].

Theorem 3.9. *Choose* $\mathrm{X}, \mathrm{Y} \in T\mathbb{R}^2$. *A length minimising bounded curvature path in* $\Gamma(\mathrm{X},\mathrm{Y})$ *is either a* CCC *path having its middle component of length greater than* π *or a* CSC *path. Some of the circular arcs or line segments can have zero length.*

Proof. Choose $\mathrm{X}, \mathrm{Y} \in T\mathbb{R}^2$ and consider a fragmentation for $\gamma \in \Gamma(\mathrm{X},\mathrm{Y})$. If γ is not a *cs* path then by applying Lemma 2.14 to each fragment we obtain a *cs* path in $\Gamma(\mathrm{X},\mathrm{Y})$ shorter than γ. We conclude that γ is not a length minimiser in $\Gamma(\mathrm{X},\mathrm{Y})$. If γ is a *cs* path of complexity greater than or equal to 4, by Corollary 3.8 we conclude that γ is not a length minimiser in $\Gamma(\mathrm{X},\mathrm{Y})$. If the complexity of γ is exactly 3 then by Corollary 3.5 we have that SCS and CCS (or SCC) paths are not length minimisers. By applying Proposition 3.6 we conclude the proof. □

The paths in the previous result are called *Dubins paths* in honour of Lester Dubins who proved Theorem 3.9 for the first time in 1957 in [14].

4. Remarks on generalisations

By applying the Pontryagin maximum principle to a time optimal control system F. Monroy-Pérez in [28] characterised the length minimisers in 2-dimensional homogeneous spaces of constant curvature. D. Mittenhuber also using an argument with control theoretical flavour obtained Dubins result in the hyperbolic 2-space.

Notice all the definitions in this work can be adapted for other surfaces. In particular, without much effort Lemma 2.9 and Lemma 2.8 can be adapted to paths in 2-dimensional homogeneous spaces of constant curvature.

4.1. **Remarks on the hyperbolic case.** Here Lemma 2.14 is a little bit more complicated to prove. But otherwise exactly the same approach works. The paths of constant curvature in the hyperbolic plane are semicircles orthogonal to the real axis including the orthogonal upper half lines. Notice in this case we cannot rescale to only deal with $\kappa = 1$ since rescaling changes the underlying curvature. So we need to always work with arcs of circles of appropriate fixed radius depending on the choice of κ.

4.2. **Remarks on the elliptic case.** For the positive curvature case we propose to work on a 2-sphere. Clearly there is a scaling issue to be addressed. This is due to the ratio between the curvature bound and the radius of the sphere. An identical method as the one employed for the euclidean or hyperbolic case (see Lemma 2.14) does not work here. Due to the geometry of the sphere, the length of a fragment in between L_1 and L_2 as in Figure 6 (but for the spherical case) could eventually be shorter than the projected path in between L_1 and L_2. This situation can be easily overcome by considering a sequence of replacements.

References

1. P. K. Agarwal, P. Raghavan, and H. Tamaki. Motion planning for a steering-constrained robot through moderate obstacles. In Proc. 27th Annu. ACM Sympos. Theory Comput., pages 343352, 1995.
2. R. P. Anderson and D. Milutinović, On the Construction of Minimum-Time Tours for a Dubins Vehicle in the Presence of Uncertainties, ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 137(3), 2015.
3. J. Ayala, Length minimising bounded curvature paths in homotopy classes, Topology and its Applications, v.193:140-151, 2015.
4. J. Ayala and J.H. Rubinstein, Non-uniqueness of the Homotopy Class of Bounded Curvature Paths (2014) arXiv:1403.4911 [math.MG].
5. J. Ayala and J.H. Rubinstein, The Classification of Homotopy Classes of Bounded Curvature Paths (2014) arXiv:1403.5314v2 [math.MG]. (to appear in the Israel Journal of Mathematics).
6. D. Balkcom and M. T. Mason, Time optimal trajectories for bounded velocity differential drive vehicles, International Journal of Robotics Research 21(3), 2002.
7. W. Blaschke, K. Reidemeister and G. Thomsen. Vorlesungen ber Differentialgeometrie und geometrische Grundlagen von Einsteins Relativittstheorie: Elementare Diifferentialgeometrie. Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen mit besonderer Bercksichtigung der Anwendungsgebiete. J. Springer, 2 edition, 1924.
8. M. Brazil, P.A. Grossman, D.A. Thomas, J.H. Rubinstein, D. Lee, N.C. Wormald, Constrained Path Optimisation for Underground Mine Layout, The 2007 International Conference of Applied and Engineering Mathematics (ICAEM07), London, (2007),856-861

9. X.-N. Bui, P. Soueres, J.-D. Boissonnat, and J.-P. Laumond. The Shortest Path Synthesis for Non-holonomic Robots Moving Forwards. Technical Report 2153, INRIA, Nice-Sophia-Antipolis, 1994.
10. J. Cantarella, J. H. G. Fu, R. B. Kusner, J. M. Sullivan Ropelength criticality, Geometry & Topology 2014.
11. A. Chang, M. Brazil, D.A. Thomas, J.H. Rubinstein, Curvature-constrained directional-cost paths in the plane. Journal of Global Optimization, Volume 53 Issue 4, August (2012), 663-681.
12. J. Diaz and J. Ayala Dubins Explorer: A software for bounded curvature paths, `http://joseayala.org/dubins_explorer.html`, 2014.
13. I. S. Dolinskaya and A. Maggiar, Time-optimal trajectories with bounded curvature in anisotropic media. The International Journal of Robotics Research 31 (14), 1761-1793.
14. L.E. Dubins, On Curves of Minimal Length with Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents, American Journal of Mathematics 79 (1957), 139-155.
15. L.E. Dubins, On Plane Curve with Curvature, Pacific J. Math. Volume 11, Number 2 (1961), 471-481.
16. O. Durumeric, Local structure of ideal shapes of knots. Topology and its Applications Volume 154, Issue 17, 15 September 2007, Pages 30703089.
17. T. El Khatib, Thickening fields for curvature-constrained paths, Masterarbeit, TU Berlin 2014.
18. S. Eriksson-Bique, D. Kirkpatrick, and V. Polishchuk, Discrete Dubins Paths. ArXiv e-prints, 2012, 1211.2365.
19. J. A. Goodman, S. J. Purkis, S. R. Phinn, Coral Reef Remote Sensing: a guide for mapping, monitoring and management. Springer Science and Business Media, Apr 18, 2013 - Technology and Engineering.
20. H. H. Johnson. An application of the maximum principle to the geometry of plane curves. Proceedings of the American Mathematical Society, 44(2):432- 435, 1974.
21. V. Jurdjevic, Geometric control theory, Cambridge Studies in Adv. Math., vol. 52, Cam. Univ. Press, Cambrdige; 1997
22. D. Kirszenblat, Dubins Networks, Master Thesis, University of Melbourne 2012.
23. S. M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, U.K. 2006.
24. J. Le Ny, J. Feron, E. Frazzoli, On the curvature-constrained traveling salesman problem. IEEE Trans. Autom. Control, in press.
25. L. B. Leopold and M. G. Wolman, Rivers Meanders, Geological Society of America Bulletin v. 71 no. 6 p. 769-793.
26. A. A. Markov. Some examples of the solution of a special kind of problem on greatest and least quantities. Soobshch. Karkovsk. Mat. Obshch., 1:250-276, 1887.
27. D. Mittenhuber, Dubins problem in the hyperbolic plane using the open disc model, in Geometric control and non-holonomic mechanics (Mexico City, 1996), CMS Conf. Proc., vol. 25, Amer. Math. Soc., Providence, RI, 1998, pp 115-152.
28. F. Monroy-Pérez. Non-Euclidean Dubins' problem. Journal of dynamical and control systems, 4(2):249-272, 1998.
29. R.N. Murray, Z.X. Li, and S.S. Sastry, A mathematical introduction to robotic manipulation, CRC Press, Boca Raton, FL; 1994.
30. G. Pestov, V. Ionin, On the Largest Possible Circle Imbedded in a Given Closed Curve, Dok. Akad. Nauk SSSR, 127:1170-1172, 1959. In Russian.
31. J. A. Reeds and L. A. Shepp, Optimal paths for a car that goes both forwards and backwards, Pacific Journal of Mathematics, vol. 145, no. 2, pp. 367?393, 1990.
32. E. Schmidt, uber das Extremum der Bogenlnge einer Raumkurve bei vorgeschriebenen Einschrnkungen ihrer Krmmung. Sitzungsberichte der Preussischen Akademie der Wissenschaften, pages 485-490, 1925.
33. A. M. Shkel, V. Lumelsky, Classification of the Dubins set. Robot. Auton. Syst. 2001, 34, 179202.
34. Y. Chitour and M. Sigalotti, Dubins problem on surfaces I: nonnegative curvature, The Journal of Geometric Analysis Volume 15, Number 4, 2005
35. M. Sigalotti and Y. Chitour, Dubins problem on surfaces II: nonpositive curvature, Siam J. Control Optim. Vol. 45, No. 2, pp. 457482.

36. H. J. Sussmann, Shortest 3-dimensional paths with a prescribed curvature bound. Proceedings of the 34th IEEE Conference on Decision and Control, pp. 3306-3312, 1995.

FIA, Universidad Arturo Prat, Iquique, Chile
E-mail address: jayalhoff@gmail.com

Department of Mathematics and Statistics, University of Melbourne Parkville, VIC 3010 Australia
E-mail address: d.kirszenblat@student.unimelb.edu.au

Department of Mathematics and Statistics, University of Melbourne Parkville, VIC 3010 Australia
E-mail address: rubin@ms.unimelb.edu.au

Bibliography

[1] Amenta, N., & Ziegler, G.M. (1996), *Deformed Products and Maximal Shadows of Polytopes*, Advances in Discrete and Computational Geometry, Amer. Math. Soc., 57–90.

[2] Anderson, D.R., Sweeney, D.J., & Williams, T.A. (2019), *An Introduction to Management Science: Quantitative Approaches to Decision Making*, Cengage.

[3] Appleby, J.S., Blake, D.V., & Newman, E.A. (1961), *Techniques for producing School Timetables on a Computer and their Application to other Scheduling Problems*, The Computer Journal, 3:237.

[4] Ayala, J., Kirszenblat, D., & Rubinstein, J.H. (2018), *A geometric approach to shortest bounded curvature paths*, Communications in Analysis and Geometry, 26(4).

[5] Broder, S. (1964), *Final Examination Scheduling*, Communications of the ACM, 7(8): 494.

[6] Burton, B.A. (2011), *The Pachner graph and the simplification of 3-sphere triangulations*, In Proceedings of the twenty-seventh annual symposium on Computational geometry (SoCG '11). ACM, New York, NY, USA: 153–162.

[7] Brazil, M., Graham, R.L., Thomas, D.A., & Zachariasen, M. (2014), *On the history of the Euclidean Steiner tree problem*, Arch. Hist. Exact Sci. 68:327.

[8] Bertsimas, D., & Tsitsiklis, J.N. (1997), *Introduction to Linear Optimization*, Athena Scientific.

[9] Bremner, D., & Lars Schewe (2011), *Edge-Graph Diameter Bounds for Convex Polytopes with Few Facets*, Experimental Mathematics, 20:3, 229–237.

[10] Bruckner, M. (1909), *Über die Ableitung der allgemeinen Polytope und die nach Isomorphismus Verschiedenen Typen der allgemeinen Achtzell (Oktatope)*, Verh. Nederl. Akad. Wetensch. Afd. Natuurk Sect. I 10, No. 1.

[11] *Canonical Labeling.* From Wolfram MathWorld, `mathworld.wolfram.com/CanonicalLabeling.html`.

[12] Chang, S.Y., & Murty, K.G. (1989), *The steepest descent gravitational method for linear*

programming, Discrete Applied Mathematics, 25(3): 211–239.

[13] Chvátal, V. (2003), *Linear Programming*, Freeman.

[14] Cipra, B.A. (2000), *The best of the 20th century: Editors name Top 10 Algorithms*, SIAM News. 33.

[15] Conforti, M. (2016), *Integer Programming*, Springer International PU.

[16] Dantzig, G.B. (1963), *Linear Programming and Extensions*, Princeton University Press, Princeton.

[17] De Loera, J.A, Rambau, J., & Francisco, S. (2010), *Triangulations: Structures for Algorithms and Applications*, Springer.

[18] De Loera, J.A., & Onn, S. (2006), *All Linear and Integer Programs Are Slim 3-Way Transportation Programs*, SIAM Journal on Optimization 17(3): 806–821.

[19] De Loera, J.A., Kim, E.D., Santos, F., & Onn, S. (2009), *Graphs of transportation polytopes*, J. Comb. Theory, Ser. A 116(8): 1306–1325.

[20] Dubins, L.E. (1957), *On Curves of Minimal Length with Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents*, American Journal of Mathematics 79: 139–155.

[21] Du, D.Z., & Hwang, F.K. (1983), *A New Bound for the Steiner Ratio*, Trans. Amer. Math. Soc., 278: 137–48.

[22] Gilbert, E.N. (1967), *Minimum Cost Communication Networks*, Bell System Technical Journal, 46: 2209–2227.

[23] Grünbaum, B., & Sreedharan, V.P. (1967), *An enumeration of simplicial 4-polytopes with 8 vertices*, J. Combinatorial Theory 2: 437–465.

[24] Grünbaum, B. (2003), *Convex Polytopes*, Springer.

[25] Hazan, E., & Megiddo, N. (2007), *The "Arrangement Method" for Linear Programming Is Equivalent to the Phase-One Method*, IBM Research Report.

[26] Kalai, G. (1988), *A simple way to tell a simple polytope from its graph*, Journal of Combinatorial Theory, Series A, 49(2): 381–383.

[27] Kalai, G., & Kleitman, D.J. (1992), *A quasi-polynomial bound for the diameter of graphs of polyhedra*, Bulletin Amer. Math. Soc., 26: 315–316.

[28] Kaya, C.Y. (2017), *Markov-Dubins path via optimal control theory*, Computational Optimization and Applications, 68(3): 719–747.

[29] Kim, E.D., & Santos, F. (2010), *An Update on the Hirsch Conjecture*, Jahresber. Dtsch.

Math. Ver., 112: 73.

[30] Kirszenblat, D., Sirinanda, K., Brazil, M., Grossman, P., Rubinstein, J.H., & Thomas, D. (2016), *Minimal curvature-constrained networks*, Journal of Global Optimization.

[31] Kirszenblat, D., Hill, B., Mak-Hau, V., Moran, B., Nguyen, V., Novak, A. (2017), *Using column generation to solve an aircrew training timetabling problem*, 22nd International Congress on Modelling and Simulation, Hobart, Tasmania, Australia, 3–8 December.

[32] Klee, V., & Walkup., D.W (1967), *The d-step conjecture for polyhedra of dimension* $d < 6$, Acta Math., 117: 53–78.

[33] Klee, V., & Minty, G.J. (1972), *How good is the simplex algorithm?* In Shisha, Oved. Inequalities III (Proceedings of the Third Symposium on Inequalities held at the University of California, Los Angeles, Calif., September 1–9, 1969, dedicated to the memory of Theodore S. Motzkin). New York-London: Academic Press. pp. 159–175.

[34] Koltun, V. (2007), *The arrangement method for linear programming*, manuscript, `http://theory.stanford.edu/?vladlen/lp.pdf`

[35] Koranne, S. (2014), *Practical Computing on the Cell Broadband Engine*, Springer.

[36] Lutz, F.H. (1999), *Triangulated Manifolds with Few Vertices and Vertex-Transitive Group Actions*, Ph.D. Thesis, Technische Universität Berlin.

[37] Markov., A.A. (1887), *Some examples of the solution of a special kind of problem on greatest and least quantities*, Soobshch. Karkovsk. Mat. Obshch., 1: 250–276.

[38] Matousek, J., & Gärtner, B. (2007), *Understanding and Using Linear Programming*, Springer.

[39] Matschke, B., Santos, F., & Weibel, C. (2015), *The width of five-dimensional prismatoids*, Proc. London Math. Soc. 110(3): 647–672.

[40] Matveev, S. (2003), *Algorithmic topology and classification of 3-manifolds*, Algorithms and Computation in Mathematics, 9, Springer, Berlin.

[41] Melzak, Z.A. (1961), *On the Problem of Steiner*, Canadian Mathematical Bulletin, 4(2): 143–148.

[42] Polya, G. (1954), *Induction and Analogy in Mathematics*, Princeton University Press.

[43] Santos, F. (2012), *A Counterexample to the Hirsch Conjecture*. Annals of Mathematics, 176(1): 383–412.

[44] Smale, S. (1998), *Mathematical Problems for the Next Century*, Mathematical Intelligencer, 20(2): 7–15.

[45] Walkup, D. (1978), *The Hirsch Conjecture Fails for Triangulated 27-Spheres*, Mathematics

of Operations Research, 3(3), 224–230.

[46] Ziegler, G.M. (2007), *Lectures on Polytopes*, Springer.

[47] Ziegler, G.M. (2012), *Who solved the Hirsch conjecture?* Documenta Mathematica Extra Volume: Optimization Stories, 75–85.

[48] Zionts, S. (1969), *The criss-cross method for solving linear programming problems*, Management Science 15: 426–445.

Author/s:
Kirszenblat, David

Title:
Topics in optimisation

Date:
2018

Persistent Link:
http://hdl.handle.net/11343/221727